ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Διαδικτυακή εφαρμογή παρακολούθησης, οπτικοποίησης και διαμερισμού επιστημομετρικών δεδομένων προσωπικού και τμημάτων επιστήμης και μηχανικής υπολογιστών στην Ελλάδα»

Του φοιτητή
Γιατσίδης Μάριος
Αρ. Μητρώου: 175112

Επιβλέπων
Στέφανος Ουγιάρογλου
Επίκουρος Καθηγητής

Ημερομηνία 10-01-2024

Τίτλος Π.Ε. Διαδικτυακή εφαρμογή παρακολούθησης, οπτικοποίησης και διαμερισμού επιστημομετρικών δεδομένων προσωπικού και τμημάτων επιστήμης και μηχανικής υπολογιστών στην Ελλάδα

Κωδικός Π.Ε. 22346

Ονοματεπώνυμο φοιτητή Μάριος Γιατσίδης

Ονοματεπώνυμο εισηγητή Στέφανος Ουγιάρογλου

Ημερομηνία ανάληψης Π.Ε. 28-11-2022

Ημερομηνία περάτωσης Π.Ε. 10-01-2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στους ανθρώπους που είναι στην καρδιά μου»

Πρόλογος

Τα περισσότερα μέλη ακαδημαϊκού προσωπικού με αντικείμενο την Επιστήμη και τη Μηχανική Υπολογιστών καθώς και την επιστήμη του Ηλεκτρονικού και Ηλεκτρολόγου Μηχανικού, συντηρούν ένα δημόσιο profile στην βιβλιογραφική βάση google scholar που αφορά επιστημομετρικές μετρήσεις της επίδοσης ερευνητών. Παρόλο που τα δεδομένα είναι ανοιχτά προς όλους, το πρόβλημα είναι ότι η έλλειψη διαφάνειας και ενοποιημένης προσέγγισης στην εκτίμηση της επιστημονικής δραστηριότητας δυσχεραίνει την αξιολόγηση και τον συγκριτικό έλεγχο. Η αποσπασμένη φύση των πληροφοριών απαιτεί από τους χρήστες, είτε αυτοί είναι ερευνητές είτε καθηγητές, να επιχειρούν εντατική έρευνα και συναρμογή δεδομένων. Αυτή η διαδικασία είναι χρονοβόρα και περίπλοκη, με αποτέλεσμα να περιορίζεται η ικανότητα των χρηστών να εξάγουν συνολικά συμπεράσματα από τα επιστημομετρικά δεδομένα.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στη δημιουργία ενός API και μιας web εφαρμογής για την ανάκτηση και οπτικοποίηση επιστημονικών δεδομένων στον τομέα της Επιστήμης, της Μηχανικής Υπολογιστών, του Ηλεκτρονικού και Ηλεκτρολόγου Μηχανικού. Το τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων συντηρεί έναν μηχανισμό άντλησης και αποθήκευσης των επιστημονικών δεδομένων. Ενσωματώνοντας το, η εργασία αποκτά πρόσβαση σε επιστημονικά στοιχεία του ακαδημαϊκού προσωπικού που δραστηριοποιείται σε πανεπιστημιακά τμήματα που ειδικεύονται στις ανωτέρω αναφερόμενες επιστήμες. Ο βασικός στόχος της εργασίας είναι η προσφορά μιας ολοκληρωμένης λύσης για τη διαχείριση και αξιοποίηση των επιστημονικών δεδομένων, επιτρέποντας στους ερευνητές να παρακολουθούν, να οπτικοποιούν και να συγκρίνουν την επιστημονική δραστηριότητα σε πανεπιστημιακά τμήματα.

«Web application for monitoring, visualizing and distributing scientometric data of computer science and engineering departments and staff in Greece»

«Giatsidis Marios»

Abstract

The present thesis focuses on creating an API and a web application for retrieving and visualizing scientific data in the fields of Science, Computer Engineering, Electronics, and Electrical Engineering. The Department of Computer and Electronic Systems Engineering maintains a mechanism for fetching and storing scientific data. By integrating this, the work gains access to scientific data from academic personnel active in university departments specializing in the aforementioned disciplines. The main objective of the thesis is to provide a comprehensive solution for managing and utilizing scientific data, allowing researchers to monitor, visualize, and compare scientific activity across university departments.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους ανθρώπους που ήταν δίπλα μου και την οικογένεια μου που με στήριξε από την αρχή.

Περιεχόμενα

Πρόλογος iv
Περίληψη v
Abstract vi
Ευχαριστίες vii
Περιεχόμενα viii
Κατάλογος Εικόνων x
Κατάλογος Πινάκων xi
Συντομογραφίες xii
Κεφάλαιο 1ο: Εισαγωγή 1
1.1 Επιστημομετρία 1
1.2 Μετρικές αξιολόγησης επιστημονικού έργου 1
1.3 Διαδικτυακοί τόποι επιστημομετρίας 4
1.4 Ο ακαδημαϊκός χάρτης πανεπιστημιακών τμημάτων Πληροφορικής της Ελλάδος 5
1.5 Κίνητρο 7
1.6 Συνεισφορά 8
1.7 Οργάνωση της εργασίας 9
Κεφάλαιο 2ο: Δεδομένα και δημιουργία βάσης δεδομένων 10
2.1 Πρωτογενή δεδομένα από το Google Scholar 10
2.2 Ανάκτηση δεδομένων από το Google Scholar 10
2.3 Η Βάση δεδομένων 12
Κεφάλαιο 3ο: Τεχνολογίες 14
3.1 Εισαγωγή 14
3.2 Frontend 14
3.2.1 React 14
3.2.2 React Router 18
3.2.3 Chart.js 2 18
3.2.4 Redux toolkit 19
3.2.5 MUI (Material-UI) 20
3.3 Backend 21
3.3.1 Node.js 21

3.3.2 Express 23
3.3.3 Sequelize 24
3.3.4 Zod 26
3.3.5 Swagger 27
3.4 Typescript 28
Κεφάλαιο 4ο: Σχεδίαση και Υλοποίηση του HellenicCSResearch 30
4.1 Λειτουργικές απαιτήσεις 30
4.2 Αρχιτεκτονική της εφαρμογής 31
4.3 Υλοποίηση του backend 31
4.4 Υλοποίηση του frontend 39
4.5 Github repository 51
Κεφάλαιο 5ο: Παρουσίαση του HellenicCSResearch 53
5.1 Αξιολόγηση ακαδημαϊκού προσωπικού 53
5.2 Αξιολόγηση τμημάτων 57
5.3 Open API 60
Κεφάλαιο 6ο: Συμπεράσματα και Μελλοντικές επεκτάσεις 63
6.1 Συμπεράσματα 63
6.2 Μελλοντικές επεκτάσεις 63
ΒΙΒΛΙΟΓΡΑΦΙΑ 64

Κατάλογος Εικόνων

Εικόνα 2.1: Google Scholar Profile 10
Εικόνα 2.2: Web scraping στο Google Scholar 11
Εικόνα 2.3: Web scraping στο Google Scholar – pagination 11
Εικόνα 2.4: Google Scholar, άρθρο χωρίς χρονολογία δημοσίευσης 12
Εικόνα 2.5: Database – ER Diagram 12
Εικόνα 3.1: Σύγκριση HTML & JS vs JSX 15
Εικόνα 3.2: Virtual DOM re-render process 17
Εικόνα 4.1: Διάγραμμα αρχιτεκτονικής της εφαρμογής 31
Εικόνα 4.2: Server class – Properties & Constructor 32
Εικόνα 4.3: Server class – listen method 33
Εικόνα 4.4: Init Swagger Docs API 33
Εικόνα 4.5: Server class – routes method 34
Εικόνα 4.6: Server class – dbConnect method 35
Εικόνα 4.7: Connect to DB via Sequelize 35
Εικόνα 4.8: Server class – middlewares method 36
Εικόνα 4.9: Database Model Associations in Sequelize 36
Εικόνα 4.10: Creating a Publication Model in Sequelize 37
Εικόνα 4.11: Routers Declaration for Academic Staff Positions 38
Εικόνα 4.12: Routers Declaration for Academic Staff Positions 38
Εικόνα 4.13: Routers Declaration for Academic Staff Positions 38
Εικόνα 4.14: HellenicCSResearch – Generic Response Object 39
Εικόνα 4.15: App Component 40
Εικόνα 4.16: The App Store 41
Εικόνα 4.17: Filter Slice 42
Εικόνα 4.18: Filter Slice 43
Εικόνα 4.19: Paths Component 44
Εικόνα 4.20: Filter Component 45
Εικόνα 4.21: PositionCheckboxes Component 46
Εικόνα 4.22: Chart Options 47
Εικόνα 4.23: PositionsPieChart - UseEffects 48
Εικόνα 4.24: PositionsPieChart – chartsData 49
Εικόνα 4.25: DeprtmentDataTable Component 50
Εικόνα 4.26: DeprtmentDataTable Component – UseEffects 51
Εικόνα 5.1: Citation Page without Filters 54
Εικόνα 5.2: Citation Page with Department filter 55
Εικόνα 5.3: Academic Staff Positions Chart 55
Εικόνα 5.4: Academic Staff tables 56
Εικόνα 5.5: Academic Staff tables – Filter 56
Εικόνα 5.6: Research Per Year Chart 57
Εικόνα 5.7: Departments Page without Filters 58
Εικόνα 5.8: Departments Page with Academic Positions filter 59
Εικόνα 5.9: Departmental Academic Positions Charts: Citations & Publications 59
Εικόνα 5.10: Departmental Academic Positions Charts Filtered Positions - Citations & Publications 60
Εικόνα 5.11: Department Research Chart 60

Εικόνα 5.12: Open API Page..... 61

Κατάλογος Πινάκων

Πίνακας 1.1: Τμήματα Πανεπιστημίων Πληροφορικής στο HellenicCSResearch..... 6

Συντομογραφίες

ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
PWA	Progressive Web APPs
I/O	Input/Output

Κεφάλαιο 1ο: Εισαγωγή

1.1 Επιστημομετρία

Η επιστημομετρία, ως επιστημονική περιοχή, αποτελεί έναν πολύτοπο τομέα που εξερευνά την επιστημονική έρευνα και τις επιδόσεις των επιστημονικών κοινοτήτων. Πρόκειται για ένα πεδίο που αναπτύσσεται με έντονο ρυθμό, εξαιτίας της διαρκούς εξέλιξης των τεχνολογιών πληροφορικής και των αναλυτικών μεθόδων. Η επιστημομετρία αποτελεί έναν συνδυασμό της επιστημονικής έρευνας και της μετρικής, με σκοπό την κατανόηση της εξέλιξης και της επίδρασης της επιστημονικής έρευνας. Η χρήση μετρικών δεικτών αποτελεί ζωτικό στοιχείο της επιστημομετρίας και μας επιτρέπει να αξιολογήσουμε την παραγωγή και την επίδραση της έρευνας. Η λέξη "Επιστημομετρία" προέρχεται από τις λέξεις "επιστήμη" και "μέτρο", αναφερόμενη στο μέτρημα και την ανάλυση των επιστημονικών εργασιών. Στην ουσία, αποτελεί έναν αναλυτικό κλάδο της επιστήμης που ασχολείται με τη συλλογή, ανάλυση και ερμηνεία των επιστημονικών δεδομένων. Η επιστημομετρία αντιμετωπίζει προκλήσεις και προοπτικές στην ψηφιακή εποχή. Με την αύξηση της ψηφιακής επικοινωνίας και των τεχνολογιών, οι μετρικοί δείκτες μπορούν να λάβουν περισσότερα και πιο ακριβή δεδομένα, όπως οι διαδικτυακές αναφορές και οι κοινωνικοί δείκτες. Παράλληλα, η πρόκληση των ανοικτών επιστημονικών δεδομένων εμφανίζει μια νέα προοπτική για την επιστημομετρία, διευκολύνοντας την ελεγχόμενη αναπαραγωγή και την αξιολόγηση της έρευνας. Η επιστημομετρία θα πρέπει να θεωρείται ως ένα πλήρες πρόγραμμα έρευνας, και όχι απλώς ως ένα σύνολο μεθόδων. Αυτό σημαίνει ότι πρέπει να προσεγγίζεται με σοβαρότητα και να συνδυάζει μεθοδολογία και θεωρητικές προσεγγίσεις [1]. Η επιστημομετρία μπορεί επίσης να παρέχει σημαντικές πληροφορίες για την αναζήτηση και την ποιότητα της βιβλιογραφίας ενός πεδίου, αναδεικνύοντας ποια έργα θεωρούνται κλασικά και ποια αποτελούν αναφορές καίριας σημασίας [2].

1.2 Μετρικές αξιολόγησης επιστημονικού έργου

Η αξιολόγηση του επιστημονικού έργου αποτελεί ένα καίριο κομμάτι της Επιστημομετρίας, καθώς επιτρέπει τον ποσοτικό και ποιοτικό προσδιορισμό της επίδοσης ενός επιστήμονα, μιας ομάδας εργασίας ή ενός επιστημονικού πεδίου. Οι μετρικές αξιολόγησης επιστημονικού έργου προσφέρουν αντικειμενικά εργαλεία για την κατανόηση, τη σύγκριση και τη βελτίωση της επιστημονικής δραστηριότητας.

Παρακάτω θα δούμε μερικές από τις πολλές μετρικές αξιολόγησης που υπάρχουν:

- h-index (Δείκτης h): αποτελεί ένα δείκτη επιστημονικής παραγωγικότητας και επιρροής που χρησιμοποιείται ευρέως για τη μέτρηση της επιστημονικής συνεισφοράς ενός ερευνητή. Εισήχθη από τον φυσικολόγο Jorge E. Hirsch το 2005 [3], μπορούμε να το συναντήσουμε με την ονομασία Hirsch index, και αποτελεί έναν αριθμητικό τρόπο να εκτιμηθεί το συνδυασμένο αντίκτυπο των δημοσιεύσεων και των παραθέσεών τους στην επιστημονική κοινότητα. Ορίζεται ως η μεγαλύτερη τιμή h, για την οποία υπάρχουν τουλάχιστον h δημοσιεύσεις που έχουν τουλάχιστον h παραθέσεις. Αυτό σημαίνει ότι ένας ερευνητής με h-index 10 έχει τουλάχιστον 10 δημοσιεύσεις, κάθε μία από τις οποίες έχει τουλάχιστον 10 παραθέσεις. Το h-index λαμβάνεται υπόψη τόσο την ποσότητα (αριθμός δημοσιεύσεων) όσο και την ποιότητα (αριθμός παραθέσεων) της επιστημονικής παραγωγής ενός ερευνητή. Έχει γίνει δεκτό ευρέως σε πολλούς επιστημονικούς τομείς ως ένας σχετικά αξιόπιστος δείκτης.

- G-index (Δείκτης g): είναι ένας δείκτης επιστημονικής παραγωγικότητας που παρουσιάζει μια πιο ισορροπημένη προσέγγιση στην αξιολόγηση της επιστημονικής επίδοσης ενός ερευνητή. Λαμβάνει υπόψη τόσο η ποσότητα όσο και η ποιότητα των δημοσιεύσεων και προτάθηκε από τον L. Egghe το 2006 [4]. Αν και όμοιο με το h-index, το g-index υπολογίζεται με βάση τον αριθμό των δημοσιεύσεων και τον αριθμό των παραθέσεων, ωστόσο, οι παραθέσεις λαμβάνονται περισσότερο υπόψη. Αυτό σημαίνει ότι οι δημοσιεύσεις που έχουν περισσότερες παραθέσεις έχουν μεγαλύτερη επίδραση στον υπολογισμό του g-index. Ο τύπος υπολογισμού του g-index είναι παρόμοιος με τον τύπο του h-index, με τη διαφορά ότι οι δημοσιεύσεις ταξινομούνται σε φθίνουσα σειρά ως προς τον αριθμό των παραθέσεων. Επιλέγεται ο μεγαλύτερος αριθμός, έτσι ώστε οι πρώτες g δημοσιεύσεις να συγκεντρώνουν τουλάχιστον g^2 παραθέσεις.
- R-index (Δείκτης R): προσπαθεί να λάβει υπόψη τόσο την ποσότητα όσο και την ποιότητα των δημοσιεύσεων ενός ερευνητή. Προέρχεται από την ανάλυση του h-index, του αριθμού των δημοσιεύσεων ενός ερευνητή που έχουν τουλάχιστον h παραπομπές. Στη συνέχεια, το R-Index συγκρίνει τον αριθμό των παραπομπών των h πιο παραπάνω άρθρων με τον αριθμό των υπόλοιπων άρθρων. Μπορεί να υπολογιστεί ως εξής: ταξινομούμε τις δημοσιεύσεις με φθίνουσα σειρά ως προ τον αριθμό των παραπομπών, βρίσκουμε το h-index και τέλος υπολογίζουμε το άθροισμα των παραπομπών των πρώτων h δημοσιεύσεων και το συγκρίνουμε με το σύνολο των παραπομπών. Ο R-index παρέχει μια ενδιαφέρουσα εναλλακτική προσέγγιση στην αξιολόγηση της επιστημονικής επίδοσης ενός ερευνητή, προσπαθώντας να εστιάσει περισσότερο στην ποιότητα των εργασιών του.
- AR-index (Δείκτης AR ή αλλιώς Author Rank Index): προσφέρει έναν τρόπο να μετρήσουμε τη συνολική επιρροή ενός επιστήμονα βασιζόμενοι σε διάφορες παραμέτρους, συμπεριλαμβανομένου του αριθμού των δημοσιεύσεών του, του αριθμού των φορές που έχει αναφερθεί και του κύρους των περιοδικών όπου έχει δημοσιεύσει. Η μεθοδολογία υπολογισμού του AR-Index λαμβάνει υπόψη τις διάφορες πτυχές της επιστημονικής δραστηριότητας και παρέχει μια ολοκληρωμένη ματιά στη συνεισφορά ενός επιστήμονα στον τομέα του. Πιο αναλυτικά συμπεριλαμβάνει τα εξής κριτήρια: συνολικός αριθμός σημοσιεύσεων και παραπομπών και το κύρος των περιοδικών, δηλαδή την επιρροή των περιοδικών όπου έχει δημοσιεύσει, λαμβάνοτνας υπόψη το Impact Factor και άλλες σχετικές μετρικές [5].
- Impact Factor (Παράγοντας Επιρροής): αποτελεί μια δημοφιλή μετρική στον κόσμο της επιστημονικής έρευνας, χρησιμοποιούμενη για την αξιολόγηση της επιρροής και της ποιότητας των επιστημονικών περιοδικών. Εισήχθη για πρώτη φορά από τον Eugene Garfield, ιδρυτή του Institute for Scientific Information (ISI), ο Impact Factor έχει γίνει ευρέως αποδεκτός ως ένα εργαλείο για την αξιολόγηση της επιστημονικής παραγωγής. Υπολογίζεται ως ο μέσος όρος των παραπομπών που λαμβάνει ένα περιοδικό για τα άρθρα του κατά τη διάρκεια ενός συγκεκριμένου χρονικού διαστήματος, συνήθως ενός έτους. Η μετρική αυτή παρέχει μια εκτίμηση του πόσο συχνά αναφέρονται τα άρθρα ενός περιοδικού από άλλες επιστημονικές δημοσιεύσεις και, ως εκ τούτου, καταδεικνύει το βαθμό της επιρροής που έχει ένα περιοδικό στον επιστημονικό χώρο. Παρόλο που ο Impact Factor είναι χρήσιμος για τη σύγκριση της σχετικής επιρροής μεταξύ περιοδικών, πρέπει να χρησιμοποιείται με προσοχή. Επικρίνεται γιατί αναδεικνύει κυρίως τα περιοδικά με συγκεκριμένα είδη ερευνών και δεν λαμβάνει υπόψη του άλλες μορφές επιστημονικής παραγωγής, όπως βιβλία και ανοικτής πρόσβασης άρθρα.

- Citation Count (Αριθμός Αναφορών): στην επιστημονική έρευνα είναι μία κυρίαρχη μετρική. Αντιπροσωπεύει τον συνολικό αριθμό φορών που μία δημοσίευση ή ένα έργο έχει αναφερθεί από άλλες επιστημονικές εργασίες, προσδίδοντας έτσι μια διάσταση της επιστημονικής επίδρασης και αναγνώρισης. Είναι σημαντική διότι αντιπροσωπεύει τον τρόπο με τον οποίο η κοινότητα επιστημονικών ερευνητών αναγνωρίζει και υιοθετεί ένα συγκεκριμένο έργο. Οι πολλαπλές αναφορές υποδεικνύουν όχι μόνο τη δημοφιλία, αλλά και τον επιστημονικό αντίκτυπο που μπορεί να έχει μια εργασία [6].
- Eigenfactor Score (Σκορ Eigenfactor): μια εξελιγμένη μετρική που χρησιμοποιείται για τον υπολογισμό της επιστημονικής επίδρασης των επιστημονικών περιοδικών. Δημιουργήθηκε από τους J. E. Hirsch και J. E. Bollen και βασίζεται στην αρχή των Eigenfactors από τη γραμμική άλγεβρα, προσπαθώντας να προσδιορίσει τον "κοινωνικό χάρτη" της επιστημονικής έρευνας. Το Eigenfactor Score λαμβάνει υπόψη του πολλούς παράγοντες, όπως οι παραπομπές από επιστημονικά περιοδικά και το βάρος των παραπομπών από περιοδικά υψηλής ποιότητας. Η μετρική αυτή υπολογίζεται χρησιμοποιώντας πολύπλοκους αλγόριθμους που λαμβάνουν υπόψη τη δομή του επιστημονικού δικτύου [7].
- SNIP (Source Normalized Impact per Paper): έχει σχεδιαστεί για να αξιολογεί την επιστημονική επίδραση μιας περιοδικής έκδοσης, λαμβάνοντας υπόψη τον επιστημονικό χώρο στον οποίο δημοσιεύεται. Δημιουργήθηκε από τον Henk Moed και στοχεύει στο να κανονίσει την επίδραση των περιοδικών με βάση την επιστημονική κοινότητα στην οποία εξυπηρετούν. Η SNIP υπολογίζεται ως το σημείο κοπής του αριθμού παραπομπών που λαμβάνει ένα περιοδικό, σε σχέση με τον αναμενόμενο αριθμό παραπομπών για τον ειδικό επιστημονικό τομέα του περιοδικού. Έτσι, η SNIP δίνει περισσότερο βάρος σε περιοδικά που δημοσιεύουν έργα που συχνά αναφέρονται σε εξειδικευμένες επιστημονικές κοινότητες [8].
- Citation Per Publication (Αναφορές ανά Δημοσίευση): ένας δείκτης της επιστημονικής επίδρασης που υπολογίζεται ως ο μέσος όρος των αναφορών που λαμβάνει μια δημοσίευση. Αυτή η μετρική εστιάζει στον τρόπο με τον οποίο κάθε άρθρο αντιμετωπίζεται από την επιστημονική κοινότητα και προσπαθεί να προσφέρει μια κανονισμένη μετρική της επίδρασης των δημοσιεύσεων. Ο υπολογισμός της CPP είναι: συνολικός αριθμός αναφορών διά τον συνολικό αριθμό δημοσιεύσεων που μας δείχνει την μέση αναγνωρισιμότητα και ανταποκρινότητας μιας ερευνητικής εργασίας [9].
- Journal Impact Factor (Συντελεστής Επίδρασης Περιοδικού): έναν από τους πιο κλασικούς δείκτες αξιολόγησης περιοδικών. Προτάθηκε από τον Eugene Garfield και δημοσιεύθηκε από το Ινστιτούτο Επιστημονικών Πληροφοριών (Institute for Scientific Information - ISI), που έχει εξελιχθεί αργότερα στην Clarivate Analytics. Ο JIF χρησιμοποιείται ευρέως για να αξιολογήσει τη σημασία ενός περιοδικού στην επιστημονική κοινότητα. Υπολογίζεται από τον Συνολικό αριθμό παραπομπών σε επιστημονικά άρθρα του περιοδικού διά τον συνολικό αριθμό επιστημονικών άρθρων που δημοσιευθήκαν στο περιοδικό [10].
- Article Influence Score (Σκορ Επίδρασης Άρθρου): αξιολογεί τη σημασία και την επίδραση των επιστημονικών άρθρων σε περιοδικά. Προτείνθηκε από τον Eugene Garfield. Το AIS υπολογίζεται από το ISI Web of Science και παρέχει ένα κανονισμένο μέτρο της επίδρασης ενός άρθρου, λαμβάνοντας υπόψη τον τομέα της επιστημονικής έρευνας. Ο υπολογισμός του AIS λαμβάνει υπόψη του την ποσότητα και τον τύπο των παραπομπών που λαμβάνει ένα άρθρο, καθώς και την επίδραση του περιοδικού στο οποίο δημοσιεύθηκε [11].
- Percentile in Subject Area (Ποσοστημόριο στην Επιστημονική Κατηγορία): παρέχει μια κανονισμένη αναφορά στη θέση ενός επιστημονικού έργου εντός μιας επιστημονικής

κατηγορίας. Χρησιμοποιείται για να αναδείξει τη σημασία ενός άρθρου ή ενός ερευνητικού έργου σε σχέση με άλλα έργα που εμπίπτουν στην ίδια κατηγορία. Το ποσοστημόριο υπολογίζεται από τις παραπομπές που λαμβάνει ένα έργο σε σχέση με όλα τα έργα που ανήκουν στην ίδια επιστημονική κατηγορία. Η χρήση αυτού του δείκτη βοηθά στο να αξιολογηθεί η επίδραση ενός έργου εντός του συγκεκριμένου επιστημονικού πεδίου [12].

- Open Access Percentage (Ποσοστό Ελεύθερης Πρόσβασης): μετρά το ποσοστό των επιστημονικών έργων που είναι ελεύθερα προσβάσιμα για το κοινό. Ενώ πολλά έργα απαιτούν συνδρομή ή πληρωμή για πρόσβαση, τα έργα με ελεύθερη πρόσβαση είναι διαθέσιμα δωρεάν σε οποιονδήποτε με πρόσβαση στο διαδίκτυο. Ο υπολογισμός γίνεται με τον Αριθμό ελεύθερων επιστημονικών έργων διά τον συνολικό αριθμό επιστημονικών έργων επί τις εκατό. Η μετρική αυτή αντικατοπτρίζει την τάση των επιστημονικών κοινοτήτων να προωθούν την ελεύθερη διάθεση της επιστημονικής πληροφορίας για την προώθηση της έρευνας [13].

1.3 Διαδικτυακοί τόποι επιστημομετρίας

Η δημιουργία των διαδικτυακών τόπων επιστημομετρίας αντικατοπτρίζει μια σημαντική μετάβαση στον τρόπο που η επιστημονική κοινότητα διαχειρίζεται, διανέμει, και αναζητά επιστημονική πληροφορία. Προτού την ύπαρξη αυτών των τόπων, η πρόσβαση σε επιστημονικά δεδομένα ήταν συχνά περιορισμένη σε περιοδικά, συνέδρια, και βιβλιοθήκες. Πριν από την ψηφιοποίηση, η αναζήτηση επιστημονικών πληροφοριών ήταν μια πολύπλοκη διαδικασία. Οι ερευνητές βασίζονταν κυρίως σε εκδοτικούς οίκους, περιοδικά, και βιβλιοθήκες για την πρόσβαση σε έρευνες. Η αναζήτηση ήταν συχνά χρονοβόρα και υπήρχε η ανάγκη για φυσική παρουσία σε ερευνητικά κέντρα. Η αρχή της ψηφιοποίησης των επιστημονικών πηγών άρχισε με τη δημιουργία των πρώτων ψηφιακών βάσεων δεδομένων, όπως η Medline για την ιατρική, το ERIC για την εκπαίδευση, και η IEEE Xplore για την ηλεκτρονική και ηλεκτρική μηχανική [14]. Αυτές οι πλατφόρμες παρείχαν ψηφιακή πρόσβαση σε επιστημονικά άρθρα, αλλά η αναζήτηση ήταν ακόμη περιορισμένη και υποχρεωτικά περιοριζόταν στο πλαίσιο της συγκεκριμένης βάσης δεδομένων.

Η ανάγκη για αξιόπιστες μετρικές αξιολόγησης είναι πιο κρίσιμη από ποτέ. Ενώ πολλοί επιστήμονες αναζητούν αξιόπιστα εργαλεία για την ανάλυση και τον υπολογισμό της επιστημονικής επίδοσης, υπάρχουν αρκετές ιστοσελίδες επιστημομετρίας που προσφέρουν ολοκληρωμένες λύσεις. Ας εξετάσουμε κάποιες από αυτές τις καινοτόμες πλατφόρμες και τα εργαλεία που αναδεικνύουν τη σύγχρονη επιστημονική πρόοδο:

Το Google Scholar αντιπροσωπεύει έναν σημαντικό κομμάτι στον ψηφιακό χώρο της επιστημονικής έρευνας, καθώς προσφέρει μια εντυπωσιακή πλατφόρμα για την πρόσβαση σε εκατομμύρια επιστημονικά άρθρα, βιβλία, και συνέδρια. Η ιστορία του χρονολογείται στις αρχές του 2004, όταν παρουσιάστηκε από τη Google ως απάντηση στην ανάγκη για μια καλύτερη και πιο εξελιγμένη μηχανή αναζήτησης για επιστημονικό περιεχόμενο. Έχει επιτύχει τη δημιουργία μιας πολυεκατομμυριακής βάσης δεδομένων που καλύπτει εκτενώς επιστημονική γνώση από διάφορους εκδοτικούς οίκους, από τα μεγάλα ως τα μικρότερα, παρέχει έναν αξιόπιστο τρόπο μέτρησης της επιστημονικής επίδοσης με μετρικές όπως ο δείκτης Hirsch (h-index) και ο δείκτης i10 και προσθέτοντας συνεχώς νέα χαρακτηριστικά και βελτιώνοντας την ακρίβεια και την ευκολία χρήσης κάνοντας να εξελίσσεται συνεχώς [15].

Το Scopus είναι ένα από τα κορυφαία εργαλεία για την αξιολόγηση, παρακολούθηση και αναζήτηση επιστημονικής έρευνας. Ξεκίνησε τη λειτουργία της το 2004 και αποτελεί απόπειρα της εταιρείας

Elsevier να συγκεντρώσει επιστημονικά έγγραφα και να παρέχει ένα σύνολο εργαλείων για την ανάλυση και την παρακολούθηση της επιστημονικής έρευνας. Καλύπτει ένα ευρύ φάσμα επιστημονικών πεδίων, περιλαμβάνοντας άρθρα, συνέδρια, βιβλία, περιοδικά και ενημερώνεται τακτικά με νέα έγγραφα και ερευνητικά αποτελέσματα. Με αντίθεση του Google Scholar παρέχει μετρικές όπως τον δείκτη CiteScore, SJR, SNIP [16].

Το ResearchGate αντιπροσωπεύει μια πλατφόρμα διαμοιρασμού επιστημονικής γνώσης και κοινότητας ερευνητών με παγκόσμια κάλυψη. Ξεκίνησε το 2008 από τους Δρ. Ijad Madisch, Δρ. Sören Hofmayer, και Βικτόρ Φόινεμπλάτ. Η βασική ιδέα πίσω από το ResearchGate ήταν να παρέχει στους ερευνητές έναν χώρο όπου μπορούν να αναρτήσουν τις ερευνητικές τους εργασίες, να επικοινωνήσουν με συναδέλφους και να ενισχύσουν τη συνεργασία. Εκτός από την δυνατότητα οι χρήστες να ανεβάζουν και να μοιράζονται τις ερευνητικές τους εργασίες αλλά και τα προφίλ ερευνητών που περιλαμβάνει πληροφορίες για την εκπαιδευτική και επαγγελματική του ιστορία, καθώς και τις εργασίες του, έχει την δυνατότητα δικτύωσης που τους επιτρέπει να συνδεθούν με άλλους ερευνητές, να παρακολουθούν τις εργασίες τους και να συμμετέχουν σε συζητήσεις. Μια δημοσίευση του ResearchGate μπορεί να έχει επιπτώσεις στην επιρροή του ερευνητή, με τη δυνατότητα συλλογής σχολίων, αξιολογήσεων και αναφορών από άλλους ερευνητές. Η ευρεία κοινότητα που προσφέρει το ResearchGate διευκολύνει την επικοινωνία και τη συνεργασία μεταξύ επιστημονικών ομάδων [17].

Το Web of Science (WoS) παρέχει πρόσβαση σε εκτενείς βάσεις δεδομένων με επιστημονικές δημοσιεύσεις, συνέδρια, και άλλα επιστημονικά έγγραφα. Δημιουργήθηκε από την εταιρεία Clarivate Analytics και ξεκίνησε τη λειτουργία της το 1964 με την ονομασία Science Citation Index [18]. Το WoS περιλαμβάνει το Science Citation Index, το Social Sciences Citation Index, και το Arts & Humanities Citation Index, καθιστώντας το μια ευρεία πλατφόρμα που καλύπτει ποικίλους τομείς, παρέχει δυνατότητα αναζήτησης βάσει παραθέσεων, επιτρέποντας στους ερευνητές να παρακολουθούν την επιρροή των εργασιών τους, εργαλεία για τη δημιουργία αναλυτικών βιβλιογραφιών, παρουσιάζοντας την ανάλυση των επιστημονικών επιδόσεων. Η δυνατότητά του να παρέχει στους ερευνητές μια εκτενή εικόνα του ερευνητικού τοπίου συνεισφέρει στην προώθηση της επιστημονικής έρευνας και την ανάπτυξη καινοτόμων ιδεών.

Οι διαδικτυακοί τόποι επιστημομετρίας έχουν ανατρέψει τα θεμέλια της επιστημονικής επικοινωνίας. Επιτρέπουν την ταχεία διάδοση της γνώσης, προάγοντας τη συνεργασία και την καινοτομία. Η πρόσβαση σε επιστημονικά δεδομένα έχει καταστεί πιο αποτελεσματική, ενώ η διασυνδετική έρευνα έχει λάβει ώθηση. Συνολικά, οι διαδικτυακοί τόποι επιστημομετρίας αποτυπώνει μια πορεία καινοτομίας που έχει βοηθήσει επιστημονική κοινότητα να εξελιχθεί, προωθώντας την ελεύθερη και ανοικτή πρόσβαση στη γνώση και διαμορφώνοντας το μέλλον της έρευνας.

1.4 Ο ακαδημαϊκός χάρτης πανεπιστημιακών τμημάτων Πληροφορικής της Ελλάδος

Η πληροφορική ως επιστήμη αντιμετωπίζει μια συνεχή εξέλιξη και διεύρυνση, καλύπτοντας ένα ευρύ φάσμα γνωστικών αντικειμένων. Από την επιστήμη των υπολογιστών και τη μηχανική υπολογιστών έως τις τηλεπικοινωνίες και τα πληροφοριακά συστήματα, η ποικιλομορφία αυτών των κλάδων προσφέρει πολλές επιλογές στους μελλοντικούς φοιτητές. Παρ' όλες τις διαφορές στην ονομασία και την οργάνωση των τμημάτων πληροφορικής, οι σπουδές σε αυτούς τους κλάδους παρέχουν τη βάση για την κατανόηση και την ανάπτυξη των σύγχρονων τεχνολογικών εφαρμογών.

Επιστήμη Υπολογιστών: η επιστήμη υπολογιστών ασχολείται με τη θεωρητική και πρακτική προσέγγιση των υπολογιστικών συστημάτων. Προσφέρει γνώσεις σε θέματα όπως η αλγοριθμική επίλυση προβλημάτων, η αρχιτεκτονική υπολογιστών και η επεξεργασία γλωσσών προγραμματισμού.

Μηχανική Υπολογιστών: η μηχανική υπολογιστών επικεντρώνεται στη σχεδίαση, την ανάπτυξη και τη διαχείριση του υλικού και του λογισμικού των υπολογιστών. Περιλαμβάνει μαθήματα για την αρχιτεκτονική επεξεργαστών και τα συστήματα ενσωματωμένων συστημάτων.

Τηλεπικοινωνίες: ο κλάδος των τηλεπικοινωνιών επικεντρώνεται στη μελέτη και την ανάπτυξη των δικτύων επικοινωνιών και των τεχνολογιών μετάδοσης δεδομένων. Περιλαμβάνει μαθήματα σχετικά με τα πρωτόκολλα επικοινωνίας, τις ασύρματες επικοινωνίες και τα δίκτυα των επόπτευσης.

Πληροφοριακά Συστήματα: τα πληροφοριακά συστήματα αφορούν τη σχεδίαση, την ανάπτυξη και τη διαχείριση των συστημάτων πληροφορικής σε επιχειρήσεις. Περιλαμβάνει μαθήματα για την ανάλυση απαιτήσεων, τον σχεδιασμό βάσεων δεδομένων και τη διοίκηση τεχνολογικών έργων.

Πίνακας 1.1: Τμήματα Πανεπιστημίων Πληροφορικής στο HellenicCSResearch

Πανεπιστήμια	**Τμήματα**
Πανεπιστήμιο Πατρών	Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Οικονομικό Πανεπιστήμιο Αθηνών	Τμήμα Πληροφορικής
Διεθνές Πανεπιστήμιο της Ελλάδος	Τμήμα Πληροφορικής
	Τμήμα Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών
	Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων
Ιόνιο Πανεπιστήμιο	Τμήμα Πληροφορικής
Πανεπιστήμιο Πειραιώς	Τμήμα Πληροφορικής
	Τμήμα Ψηφιακών Συσημάτων
Πανεπιστήμιο Κρήτης	Τμήμα Επιστήμης Υπολογιστών
Πανεπιστήμιο Ιωαννίνων	Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής
	Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Δυτικής Μακεδονίας	Τμήμα Πληροφορικής
	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας	Τμήμα Πληροφορικής και Τηλεπικοινωνιών
	Τμήμα Πληροφορικής με Εφαρμογές στη Βϊοιατρική
	Τμήμα Ψηφιακών Συσημάτων
	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών

	Υπολογιστών
Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης	Τμήμα Πληροφορικής
	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Μακεδονίας	Τμήμα Εφαρμοσμένης Πληροφορικής
Εθνικό Καποδιστριακό Πανεπιστήμιο Αθηνών	Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Χαροκόπειο Πανεπιστήμιο	Τμήμα Πληροφορικής και Τηλεματικής
Πανεπιστήμιο Πελοποννήσου	Τμήμα Πληροφορικής και Τηλεπικοινωνιών
	Τμήμα Ψηφιακών Συστημάτων
	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Ελληνικό Μεσσογιακό Πανεπιστήμιο	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
	Τμήμα Ηλεκτρονικών Μηχανικών
Εθνικό Μετσόβειο Πολυτεχνείο	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πολυτεχνείο Κρήτης	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Δημοκρίτειο Πανεπιστήμιο Θράκης	Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Δυτικής Αττικής	Τμήμα Ηλεκτρολόγων και Ηλεκτρονικών Μηχανικών
	Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών
Πανεπιστήμιο Αιγαίου	Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων
Πανεπιστήμιο Πατρών	Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών

1.5 Κίνητρο

Το κίνητρο πίσω από την ανάπτυξη της παρούσας πτυχιακής εργασίας απορρέει από την ανάγκη για την οργάνωση και οπτικοποίηση των ανοιχτών επιστημομετρικών δεδομένων από το Google scholar, που προέρχονται από το προσωπικό και τα τμήματα επιστήμης και μηχανικής υπολογιστών στην Ελλάδα. Οι επιστημομετρικές πληροφορίες που παρέχονται αν και αυτά τα δεδομένα είναι προσβάσιμα, η έλλειψη ενός οργανωμένου και ευέλικτου εργαλείου παρακολούθησης καθιστά δύσκολο τον εντοπισμό και την ανάκτηση συγκεκριμένων πληροφοριών όπως για τον συγγραφέα και ειδικότερα για ένα ολόκληρο τμήμα με δεκάδες προσωπικό.

Το κύριο πρόβλημα που αντιμετωπίζεται είναι η έλλειψη οργάνωσης των πληροφοριών που παρέχονται καθημερινά από το προσωπικό στο διαδικτυακό τόπο επιστημομετρίας, δημιουργώντας ένα ασαφές και δύσκολο στον εντοπισμό περιβάλλον. Η αναζήτηση για συγκεκριμένο έτος, ένα εύρος χρονολογιών, θέση/θέσεις προσωπικού ή ακόμα και για συγκεκριμένα τμήματα αποτελούν πρόκληση και καθιστά τη διαδικασία αρκετά περίπλοκη και χρονοβόρα, καθώς ο χρήστης υποχρεούται να επισκέπτεται κάθε προφίλ του προσωπικού ξεχωριστά.

Οι καθηγητές συχνά αντιμετωπίζουν δυσκολίες στον εντοπισμό συγκεκριμένων πληροφοριών σχετικά με τις δημοσιεύσεις τους, τις ερευνητικές τους εξελίξεις και τις επιδόσεις τους σε διάφορα χρονικά διαστήματα. Η ανάγκη να επισκέπτονται διάφορα προφίλ και να αναζητούν δεδομένα σε διάσπαρτες πλατφόρμες καθυστερεί την ανάκτηση της απαραίτητης ενημέρωσης και επιβραδύνει τη διαδικασία λήψης αποφάσεων για την καριέρα τους και την προώθηση της έρευνάς τους. Επιπλέον, η αποσπασματική παρουσίαση των επιστημονικών δεδομένων καθιστά δύσκολο τον πλήρη αντίκτυπο της έρευνας τους στον ευρύτερο ακαδημαϊκό κοινό.

1.6 Συνεισφορά

Στόχος της πτυχιακής εργασίας είναι η δημιουργία μιας προηγμένης διαδικτυακής εφαρμογής, της HellenicCSResearch, δίνοντας έμφαση στην ανάλυση, την οπτικοποίηση και την πρόσβαση σε δεδομένα. Επιπλέον προσφέρει ένα δωρεάν open-source RESTful API με πολλές δυνατότητες για την άντληση αυτών τον στατιστικών.

Η διαδικτυακή εφαρμογή που αναπτύχθηκε ανοίγει νέους ορίζοντες για τους χρήστες, περιλαμβάνοντας καθηγητές και άλλα ενδιαφερόμενα μέλη. Επιτρέπει την οπτικοποίηση δεδομένων, προσφέροντας εύκολη πρόσβαση και κατανόηση. Οι χρήστες επιτρέπεται να εξερευνήσουν πληροφορίες και να προβαίνουν σε αναλύσεις για τον τομέα που τους ενδιαφέρει. Η αναζήτηση για τα δεδομένα που θέλει να αναζητήσει ο εκάστοτε χρήστης γίνεται μέσω των φίλτρων τα οποία είναι τα παρακάτω:

- Εύρος χρονολογιών: επιτρέπει στην επιλογή πολλαπλών ή και ενός έτους. Επιπλέον έχει μία έξτρα επιλογή ώστε να εμφανίζει και τις δημοσιεύσεις οι οποίες δεν έχουν έτος κυκλοφορίας.
- Τμήματα: περιέχει όλα τα τμήματα Πληροφορικής της Ελλάδος. Έχει την δυνατότητα να επιλέξει όσα τμήματα επιθυμεί ο χρήστης.
- Ακαδημαϊκή θέση: Ίδια λειτουργία με το φίλτρο τμήματα αλλά είναι για τις θέσεις τις οποίες το προσωπικό ενός τμήματος μπορεί να έχει.

Η εφαρμογή είναι χωρισμένη σε δύο σελίδες. Η πρώτη σελίδα έχει να κάνει με τις παραπομπές των δημοσιεύσεων του προσωπικού. Με την χρήση των παραπάνω φίλτρων μπορεί κάποιος να αναζητήσει την δραστηριότητα καθηγητών από συγκεκριμένα τμήματα από όλη την Ελλάδα, να ψάξει για τις επιδόσεις που έχει μία ακαδημαϊκή θέση τα τελευταία 2, 5, 10 χρόνια. Μπορεί να δει κάποια στατιστικά στοιχεία από όλα τα φίλτρα που του έχει δώσει ο χρήστης, όπως ο μέσος όρος των δημοσιεύσεων/παραπομπών ανά προσωπικό. Υπάρχουν δύο πίνακες οι οποίοι είναι συνδεδεμένη μεταξύ τους και κάθε γραμμή του πίνακα δείχνει τα δεδομένα ανά χρήστη που έχει οριστεί από τα φίλτρα. Ο πρώτος πίνακας επικεντρώνεται στα γενικά αλλά και στατιστικά στοιχεία του χρήστη για τις χρονολογίες που έχουν δοθεί όπως h-index, συνολικές δημοσιεύσεις κλπ. Ο δεύτερος πίνακας έχει να κάνει με τις επιδόσεις ανά χρονιά και μπορεί είτε να δει τις παραπομπές ή τις δημοσιεύσεις ή και τα δύο μαζί. Πατώντας μια γραμμή σε έναν από τους δύο πίνακες, γίνεται η οπτικοποίηση μέσω γραφημάτων ώστε να γίνεται πιο εύκολα κατανοητή η πληροφορία από τον πίνακα με τις χρονιές.

Η δεύτερη σελίδα ασχολείται με τα τμήματα και μπορεί ο κάθε ενδιαφερόμενος να δει, να εξετάσει και να συγκρίνει τα τμήματα μεταξύ τους. Λόγω της πληροφορίας που εστιάζει η σελίδα αυτή, δεν υπάρχει το φίλτρο για τα τμήματα επειδή αυτομάτως δείχνει για όλα τα τμήματα αλλά θα πρέπει να διαλέξει ο χρήστης για ποια ακαδημαϊκή θέση. Υπάρχει και εδώ ένας αναλυτικός δυναμικός πίνακας με τα στοιχεία του κάθε τμήματος π.χ. ο μέσος όρος παραπομπών/αναφορών ανά προσωπικό του τμήματος, CV παραπομπών/αναφορών του τμήματος και άλλα παρόμοια στατιστικά. Διαλέγοντας κάποιο τμήμα, εμφανίζονται διάφορα διαγράμματα για τις παραπομπές και δημοσιεύσεις ανά ακαδημαϊκή θέση αλλά και διάγραμμα ανά προσωπικό.

Το open-source RESTful API είναι ένα πολύτιμο εργαλείο που προσφέρει όλα τα δεδομένα του προσωπικού των τμημάτων πληροφορικής της Ελλάδος που εμφανίζει το HellenicCSResearch και όχι μόνο. Ο ενδιαφερόμενος μπορεί να δει το documentation του API μέσα από σύνδεσμο που προσφέρεται μέσα στην διαδικτυακή εφαρμογή.

1.7 Οργάνωση της εργασίας

Η δομή της εργασίας χωρίζεται σε έξι κεφάλαια, και στο κείμενο αυτό θα εξετάσουμε συνοπτικά τα κύρια θεματικά κεφάλαια, χωρίς να εστιάσουμε σε λεπτομερείς αναλύσεις. Πιο συγκεκριμένα:

Στο δεύτερο Κεφάλαιο, αναλύονται τα δεδομένα και η δημιουργία της βάσης δεδομένων, τα πρωτογενή δεδομένα και ανάκτηση αυτών από το google scholar, των τεχνολογιών που χρησιμοποιήθηκαν για την υλοποίηση της βάσης, η δομή και τα στατιστικά στοιχεία της.

Στο τρίτο Κεφάλαιο, παρουσιάζονται οι τεχνολογίες και τους λόγους επιλογής τους που χρησιμοποιήθηκαν στο backend (server) και στο frontend (διαδικτυακή εφαρμογή), καθώς και το πως επηρέασαν το developer experience.

Στο τέταρτο Κεφάλαιο, εστιάζετε στον σχεδιασμό και υλοποίηση του HellenicCSResearch. Εστιάζει τις λειτουργικές απαιτήσεις, την αρχιτεκτονική της εφαρμογής που αναπτύχθηκε για την πτυχιακή εργασία, παραθέτοντας αποσπάσματα κώδικα για την κατανόηση της λειτουργικότητας του backend και frontend. Τέλος, το github repository το οποίο χρησιμοποιήθηκε για την διαχείριση του κώδικα.

Στο πέμπτο Κεφάλαιο, παρουσιάζεται το HellenicCSResearch. Επικεντρώνεται στο interface που βλέπει και επιδράει ο τελικός χρήστης, σε όλες τις σελίδες αλλά και κάποια σενάρια χρήση της εφαρμογής. Επιπλέον, γίνεται αναλυτική περιγραφή των endpoints του open RESTful API

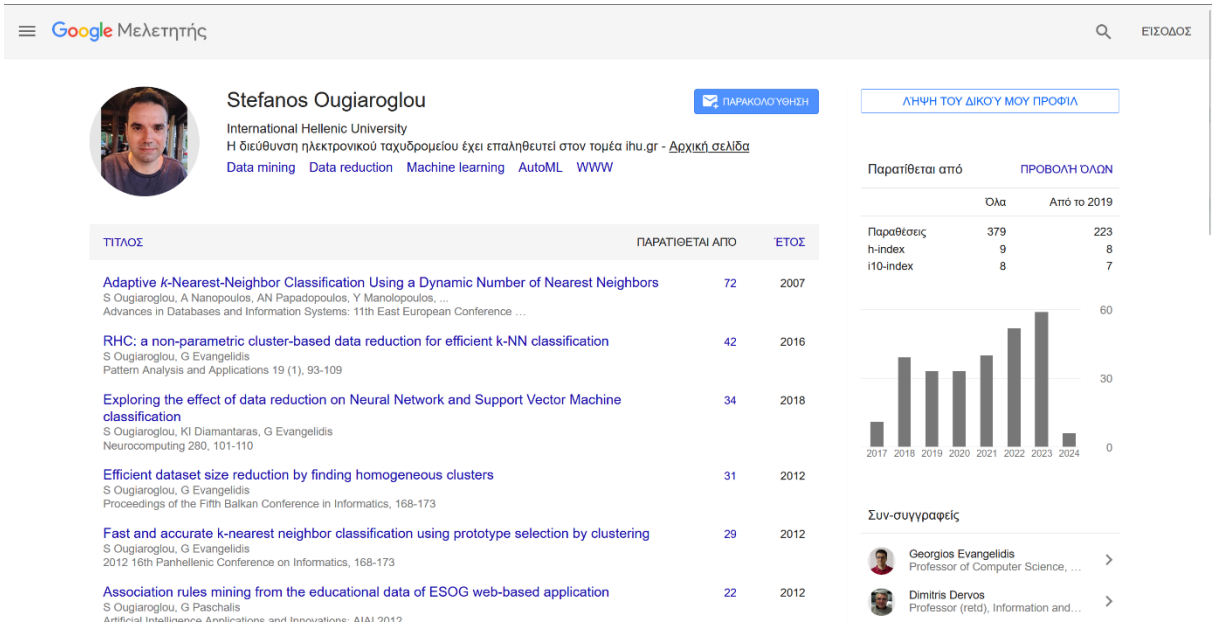
Τέλος, στο έκτο Κεφάλαιο, περιγράφονται τα συμπεράσματα, τι προσφέρει αυτή η εφαρμογή και μερικές μελλοντικές κατευθύνσεις που θα βοηθήσουν να εξελιχθεί αυτό τον τομέα που καλύπτει η πτυχιακή.

Κεφάλαιο 2ο: Δεδομένα και δημιουργία βάσης δεδομένων

2.1 Πρωτογενή δεδομένα από το Google Scholar

Το δημόσιο προφίλ στο Google Scholar αποτελεί ένα ισχυρό εργαλείο που επιτρέπει στους ερευνητές να παρουσιάσουν και να καταγράψουν την επιστημονική τους δραστηριότητα. Προσφέρει μια συγκεντρωμένη και προσβάσιμη πλατφόρμα για την προβολή των επιστημονικών τους επιτευγμάτων, ενώ παράλληλα παρέχει χρήσιμες πληροφορίες σε άλλους ερευνητές, φοιτητές και επαγγελματίες προσφέροντας ένα ολοκληρωμένο και ευέλικτο μέσο.

Ένα παράδειγμα ενός προφίλ στο Google Scholar.



Εικόνα 2.1: Google Scholar Profile

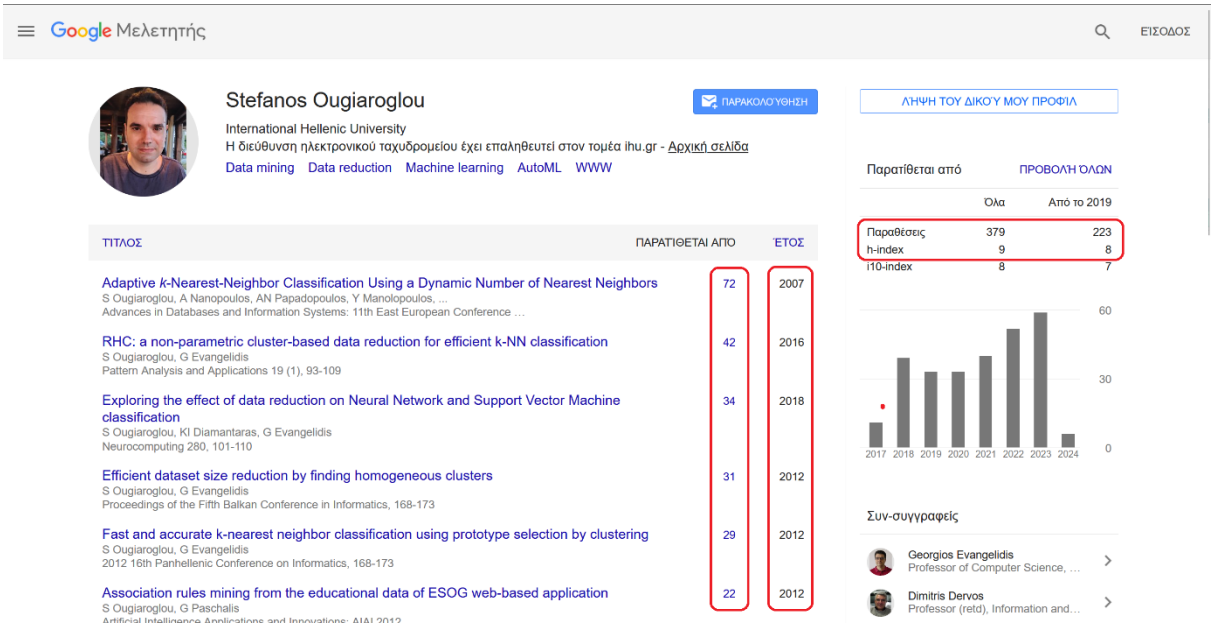
Αυτό είναι το προφίλ του Επίκουρου Καθηγητή Στέφανου Ουγιάρογλου. Όπως φαίνεται στην Εικόνα 2.1 *Σελίδα προφίλ του Google Scholar*, στην επάνω μεριά φαίνονται κάποια βασικά στοιχεία του χρήστη όπως το ονοματεπώνυμο και σε ποιο πανεπιστήμιο εργάζεται. Κάτω από αυτό υπάρχει ένας πίνακας για τις δημοσιεύσεις ο οποίος αποτελείται από τα εξής χαρακτηριστικά: τίτλο, παραπομπές και έτος. Στα δεξιά υπάρχει η στήλη με τα στατιστικά στοιχεία του εκάστοτε χρήστη και είναι χωρισμένο σε δύο κατηγορίες: στο γενικό σύνολο και τα τελευταία πέντε χρόνια. Τα στατιστικά που εμφανίζει το Google Scholar είναι για τις παραθέσεις, τον δείκτη h-index και i10-index.

2.2 Ανάκτηση δεδομένων από το Google Scholar

Η ανάκτηση δεδομένων δεν αναπτύχθηκε μέσα στα πλαίσια της πτυχιακής εργασίας αλλά η υλοποίηση έγινε από τον Επίκουρο Καθηγητή κ. Ουγιάρογλου Στέφανο. Καθώς η Google δεν έχει κάποιο open API στο Scholar ώστε να μπορεί ένα τρίτο άτομο να συλλέγει δεδομένα, έτσι έγινε η χρήση web scraping που κάνει αυτοματοποιημένη την διαδικασία ανάκτησης πληροφοριών, διευκολύνοντας τη συλλογή μεγάλων ποσοτήτων δεδομένων. Πρέπει να σημειωθεί ότι η χρήση του πρέπει να γίνεται σύμφωνα με τους όρους χρήσης της ιστοσελίδας που προορίζεται να συλλέξει δεδομένα, και με σεβασμό προς τις διατάξεις περί προστασίας δεδομένων και πνευματικής ιδιοκτησίας [19]. Πριν από την χρήση του script έπρεπε να γίνει μία προεργασία, λόγω έλλειψης

πληροφοριών που υπάρχουν στο σάιτ, απαιτούνταν η επίσκεψη στην ιστοσελίδα των τμημάτων Πληροφορικής ώστε να γίνει η καταγραφεί γενικών στοιχείων και του προσωπικού κάθε τμήματος. Επιπλέον, μία ακόμη πολύ σημαντική απουσία που έχει το Google Scholar είναι η ακαδημαϊκή θέση που χωρίς αυτήν θα χανόταν ένα μεγάλο κομμάτι συγκρίσεων και συμπερασμάτων από τα δεδομένα αυτά. Εφόσον ο κ. Ουγιάρογλου είχε το προσωπικό κάθε τμήματος, μπορούσε πλέον να κάνει web scraping μέσω ενός python script που έτρεχε ανά προφίλ.

Στην Εικόνα 2.2 *Web scraping στο Google Scholar*, όπου δείχνει με κόκκινα πλαίσια τα στοιχεία που το script βλέπει για να αντλήσει τα δεδομένα.



Εικόνα 2.2: Web scraping στο Google Scholar

Στην Εικόνα 2.3 *Web scraping στο Google Scholar – pagination*, δείχνει μία πρόκληση για το script λόγω της σελισελιδοποίησης που υπάρχει στον πίνακα με τις δημοσιεύση, που έπρεπε να κάνει πρώτα εμφάνιση όλων των άρθρων.

Εικόνα 2.3: Web scraping στο Google Scholar – pagination

Ενώ κάποιες δημοσιεύσεις δεν έχουν την χρονολογία, το script κρατάει σαν πληροφορία ότι υπάρχουν στο σύνολο και αυτές, όπως φαίνεται στην Εικόνα 2.3 *Google Scholar, άρθρο χωρίς χρονολογία δημοσίευσης*

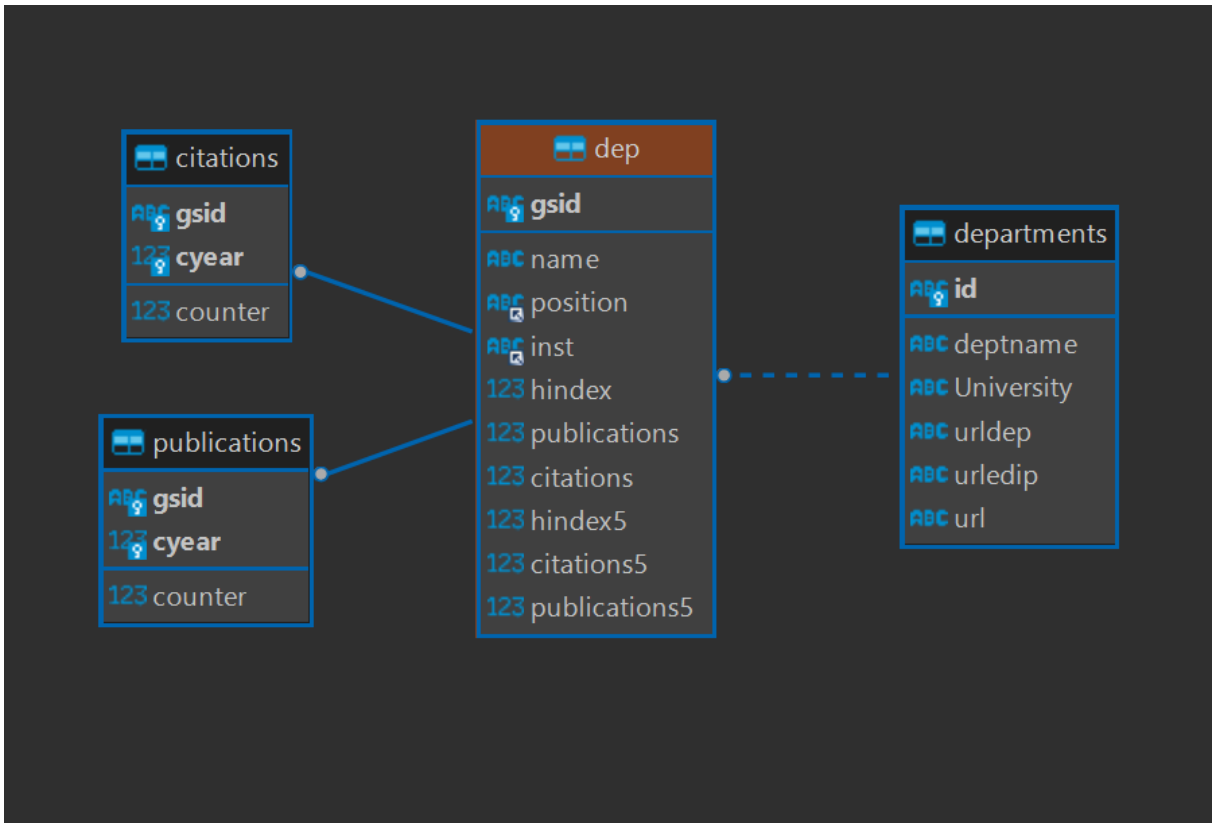
Instance-based classification using prototypes generated
S Ougiaroglou, DA Dervos, G Evangelidis

Εικόνα 2.4: Google Scholar, άρθρο χωρίς χρονολογία δημοσίευσης

Το script εφόσον τελείωνε με το web scraping, στο τέλος έκανε επεξεργασία τα δεδομένα και γινόταν σύνδεση με την βάση δεδομένων και εκτελούσε είτε create είτε update ανάλογα εάν υπήρχε το προσωπικό που γινόταν εισαγωγή στην βάση. Το συγκεκριμένο script θα τρέχει ανά ένα μήνα και θα ανανεώνει την βάση με τα πιο πρόσφατα δεδομένα που θα έχουν ανέβει στο Google Scholar.

2.3 Η Βάση δεδομένων

Η βάση περιέχει τέσσερις πίνακες με τα ονόματα: dep, departments, publications και citations:



Εικόνα 2.5: Database – ER Diagram

Ο πίνακας dep είναι ο κύριος πίνακας της βάσης, αποτελείται από τις στήλες:

- gsid (string): το πρωτεύον κλειδί του πίνακα, το οποίο είναι το id που δίνει το Google Scholar στο url του προφίλ του κάθε επιστήμονα (Google Scholar ID). Λειτουργεί και σαν ξένο κλειδί για τους πίνακες publications και citations.
- name (string): το όνομα του επιστήμονα.
- position (string): η ακαδημαϊκή θέση.
- inst (string): το id των τμημάτων (π.χ. iee@ihu), είναι επίσης ξένο κλειδί στο πίνακα departments.
- hindex (int): η μετρική h-index.
- publications (int): το σύνολο των δημοσιεύσεων.
- citations (int): το σύνολο των παραπομπών.

- hindex5 (int): η μετρική h-index για τα τελευταία πέντε χρόνια.
- publications5 (int): το σύνολο των δημοσιεύσεων για τα τελευταία πέντε χρόνια.
- citations5 (int): το σύνολο των παραπομπών για τα τελευταία πέντε χρόνια.

Για να αποφευχθεί ώστε να μην υπάρχει επαναλαμβανόμενη πληροφορία για τα δεδομένα του χρήστη έπρεπε οι δημοσιεύσεις και οι παραπομπές να γίνουν ξεχωριστοί πίνακες αλλά φυσικά δεν θα μπορούσαν να λειτουργήσουν με κύριο κλειδί μόνο το gsid, οπότε για να έχει μοναδικά κλειδιά έχουν τις δύο στήλες gsid και cyear.

Πιο αναλυτικά ο πίνακας citations:

- gsid (string): το ένα από τα δύο κλειδιά του πίνακα, το οποίο είναι το id που δίνει το Google Scholar στο url του προφίλ του κάθε επιστήμονα (Google Scholar ID).
- cyear (int): χρονολογίες που έγινε τουλάχιστον μία δημοσίευση ανά έτος από το προσωπικό με το αντίστοιχο gsid.
- counter (int): ο συνολικός αριθμός δημοσιεύσεων του έτος χ.

Και publications:

- gsid (string): το ένα από τα δύο κλειδιά του πίνακα, το οποίο είναι το id που δίνει το Google Scholar στο url του προφίλ του κάθε επιστήμονα (Google Scholar ID).
- cyear (int): το δεύτερο κλειδί του πίνακα, έχει τις χρονολογίες που έγινε τουλάχιστον μία δημοσίευση ανά έτος από το προσωπικό με το αντίστοιχο gsid.
- counter (int): ο συνολικός αριθμός παραπομπών του έτος χ.

Επειδή υπάρχουν άρθρα τα οποία δεν έχουν έτος δημοσίευσης, στον πίνακα citations, εισάγονται αλλά στην στήλη cyear με αριθμό -1.

Τέλος, ο πίνακας departments, οποίος έχει στοιχεία για τα τμήματα:

- id (string): κύριο κλειδί, το id του τμήματος
- deptname (string): το όνομα του τμήματος.
- University (string): το όνομα του πανεπιστημίου που ανήκει το τμήμα.
- urldep (string): η ιστοσελίδα του προσωπικού του τμήματος.
- urledip (string): η ιστοσελίδα του τεχνικού προσωπικού του τμήματος.
- url (string): η ιστοσελίδα του πανεπιστημίου.

Κεφάλαιο 3ο: Τεχνολογίες

3.1 Εισαγωγή

Η ανάπτυξη της πτυχιακής εργασίας βασίστηκε σε μια σειρά τεχνολογιών. Πιο συγκεκριμένα, στον τομέα του Backend, χρησιμοποιήθηκε η δημοφιλής πλατφόρμα Node.js (που αποτελεί το κύριο κομμάτι του), τα frameworks Express και Sequelize, για την HTTP επικοινωνία και για τη διαχείριση SQL queries για την συλλογή δεδομένων από την MariaDB βάση, το Swagger για documentation. Στον τομέα του Frontend, η React προσφέρει ευελιξία και περίπλοκα UI interfaces και μια μεγάλη γκάμα από βιβλιοθήκες. Για το UI, έγινε χρήση της MUI (Material-UI), η οποία προσφέρει σύγχρονο και εύκολα τροποποιήσιμο σχεδιασμό. Η επικοινωνία με το Backend έγινε με τη χρήση της RTK (Redux Toolkit), ενώ για τη δημιουργία δυναμικών γραφημάτων χρησιμοποιήθηκε το Chart.js 2, που είναι προσαρμοσμένο για το περιβάλλον της React. Ο συνδυασμός της ενσωμάτωσης με την TypeScript, στο Backend αλλά και στο Frontend κομμάτι, βοήθησε στην ανάπτυξη της πτυχιακής.

3.2 Frontend

3.2.1 React

Η React είναι μία δημοφιλείς δωρεάν και open-source frontend βιβλιοθήκη για την γλώσσα προγραμματισμού Javascript, που χρησιμοποιείται για την κατασκευή UI components και διεπαφές χρηστών.

Η δημιουργία της έγινε από το Facebook και εξακολουθεί να συντηρείται από αυτήν, μαζί με μια συνεχώς αυξανόμενη κοινότητα εταιρειών και ατόμων προγραμματιστών. Η React χρησιμοποιείται συχνά ως η βάση κώδικα για Εφαρμογές μονής σελίδας (SPA). Βρίσκεται επί του παρόντος στην κορυφή των διαγραμμάτων, όπως η χρήση της, δημοφιλία, την ικανοποίηση των προγραμματιστών και τα εργαλεία που θέλει περισσότερο η βιομηχανία [20].

Η ιδέα για το ξεκίνημα της άρχισε το 2011 όπου η εφαρμογή διαφημίσεων της Facebook άρχισε να κερδίζει μεγάλη ώθηση, με αποτέλεσμα η ομάδα προγραμματιστών μεγάλωνε και ο κώδικας μαζί, που έγινε δύσκολο να διαχειριστεί μετά από ένα σημείο. Προκειμένου να βοηθήσει να αντιμετωπιστεί το πρόβλημα το οποίο αυξανόταν σταδιακά, ο Jordan Walke δημιούργησε το FaxJS [21], ένα πρωτότυπο που έκανε αυτήν τη διαδικασία πιο αποτελεσματική. Αυτή ήταν η «γέννηση» αυτού που αργότερα εξελίχθηκε στην έκδοση της React που όλοι γνωρίζουν και χρησιμοποιούν.

Μέχρι τότε ήταν closed source αλλά όλα άλλαξαν το 2012 που η Facebook αγόρασε μια μικρή εταιρία που λεγόταν Instagram, η οποία είχε τότε μόλις 13 υπαλλήλους. Μετά την εξαγορά της, η Instagram ήθελε να υιοθετήσει πολλές από τις νέες τεχνολογίες της Facebook. Αυτό άσκησε πίεση στη μητρική εταιρεία και έτσι έκαναν την React open-source. Στο Q3 του 2013, ο Jordan Walke εισήγαγε αυτό που τώρα λέμε React και κατέστη open-source. Στην υπόλοιπη διάρκεια του έτους, η React άρχισε να κερδίζει σημαντικό κύρος καθώς προστέθηκαν σημαντικά νέα εργαλεία όπως το React Developer Tools που προστέθηκε ως επέκταση των Chrome Developer Tools [22].

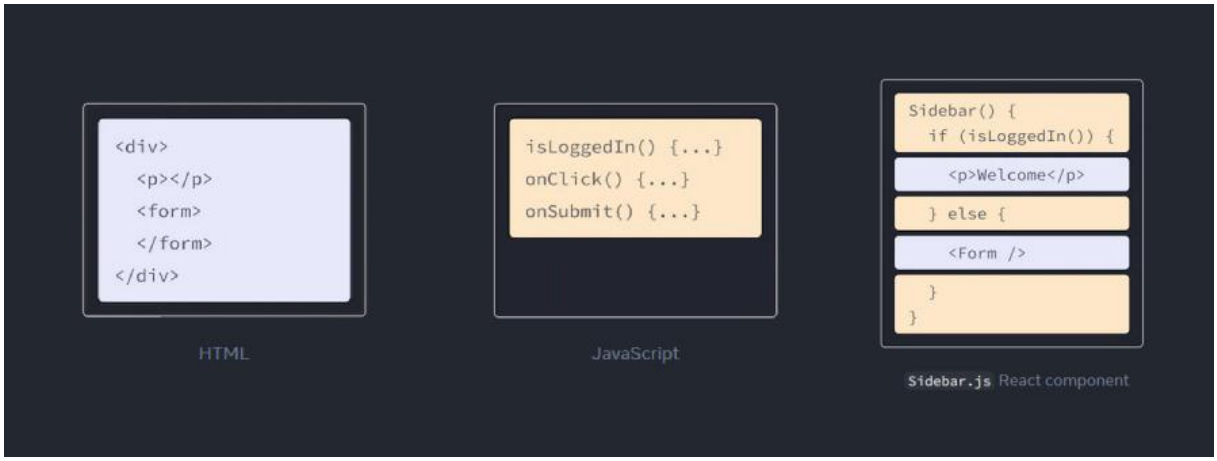
Το 2015, η React έφτασε στο σημείο όπου θεωρείται stable, όπου το Netflix δημοσίευσε ένα άρθρο υπέρ της React και έπειτα το Airbnb άρχισε να χρησιμοποιεί τη React. Ακολούθησε η εισαγωγή της React Native για iOS και Android που έγινε και αυτή open-source τον Μάρτιο [23]. Καθώς δημιουργήθηκε μια μεγάλη και πολύ ενεργή κοινότητα γύρω από τη React, είδε πολλές άλλες

βελτιώσεις και βελτιστοποιήσεις που έχουν συμβάλει ακόμη περισσότερο στην άνοδο της ως ουσιαστικό μέρος πολλών σύγχρονων λογισμικών προϊόντων και εταιρειών.

Η JSX (JavaScript Syntax Extension), είναι ένα ουσιαστικό μέρος της React. Η JSX μοιάζει πολύ με την HTML, μία γλώσσα που χρησιμοποιείται για τη δομή και την παρουσίαση περιεχομένου στον ιστό, αλλά έχει την πλήρη ισχύ της JavaScript. Ο κώδικας είναι πολύ παρόμοιος με το να γράφει κάποιος HTML στον JS κώδικα του, κάνοντας το πιο εύκολο στην κατανόηση και τη διατήρηση του. Βασικό χαρακτηριστικό του, η ικανότητα να παράγει React “elements”, που είναι τα πιο μικρά blocks κατασκευής που έχει και περιγράφουν το τι είναι να εμφανιστεί στην οθόνη. Αυτά τα elements που κάνει render η React στο DOM (Document Object Model), το οποίο αντιπροσωπεύει τη δομή των ιστοσελίδων. Η JSX υποστηρίζει JavaScript expressions, που σημαίνει ότι μπορεί να ενσωματώσει τιμές ή να καλεί μεθόδους. Η μόνη προϋπόθεση είναι να τα τυλίξουμε με διπλά άγκιστρα. Η χρήση της JSX δεν είναι υποχρεωτική.

Κάποιες διαφορές που έχει η JSX με σχέση την HTML είναι ότι όλες οι ετικέτες από τα elements πρέπει να κλείνουν. Μία ακόμη διαφορά είναι στις δηλώσεις κλάσεων ενός element, η JSX για να δηλώσει κλάσεις πρέπει να χρησιμοποιήσει το ‘className’ διότι στην JS το class δεσμευμένη λέξη. Οπότε είναι σημαντικό να κατανοήσει ο προγραμματιστής τη σύνταξη της. Στην Εικόνα 3.1 *Σύγκριση HTML & JS vs JSX*, δείχνει ένα απλό παράδειγμα για το πως να γραφτεί ένα κομμάτι λογικής από HTML και JS σε JSX.

Ενώ η JSX βελτιώνει την αναγνωσιμότητα και τη συντηρησιμότητα του κώδικά, δεν είναι native JavaScript και χρειάζεται να μεταγλωττιστεί σε JavaScript πριν μπορέσει να τρέξει σε έναν περιηγητή. Εργαλεία όπως το Babel χρησιμοποιούνται για αυτή τη μετατροπή κατά τη διάρκεια του σταδίου ανάπτυξης κώδικα.



Εικόνα 3.1: Σύγκριση HTML & JS vs JSX

Το Component είναι ένας από τους βασικούς πυρήνες της React, που κάνουν την εργασία της δημιουργίας διεπαφών χρήστη πολύ πιο εύκολη. Μια διεπαφή χρήστη έχει πολλά ξεχωριστά, επαναχρησιμοποιούμενα κομμάτια Components. Στην React υπάρχουν δύο τύποι που μπορεί να γράψει ο προγραμματιστής:

- Functional Components: είναι απλά μέθοδοι JavaScript. Αυτές οι μέθοδοι μπορεί ή να μην λαμβάνουν δεδομένα ως παραμέτρους (props).
- Class Components: είναι λίγο πιο πολύπλοκα από τα Functional Components. Τα Functional Components δεν είναι ενήμερα για τα άλλα στοιχεία του προγράμματος σας, ενώ τα Class

Components μπορούν να λειτουργούν μεταξύ τους. Μπορούμε να περάσουμε δεδομένα από το ένα σε ένα άλλο Class Components. Μπορούμε να χρησιμοποιήσουμε τις κλάσεις JavaScript ES6 για τη δημιουργία συστατικών βασισμένων σε κλάσεις στη React.

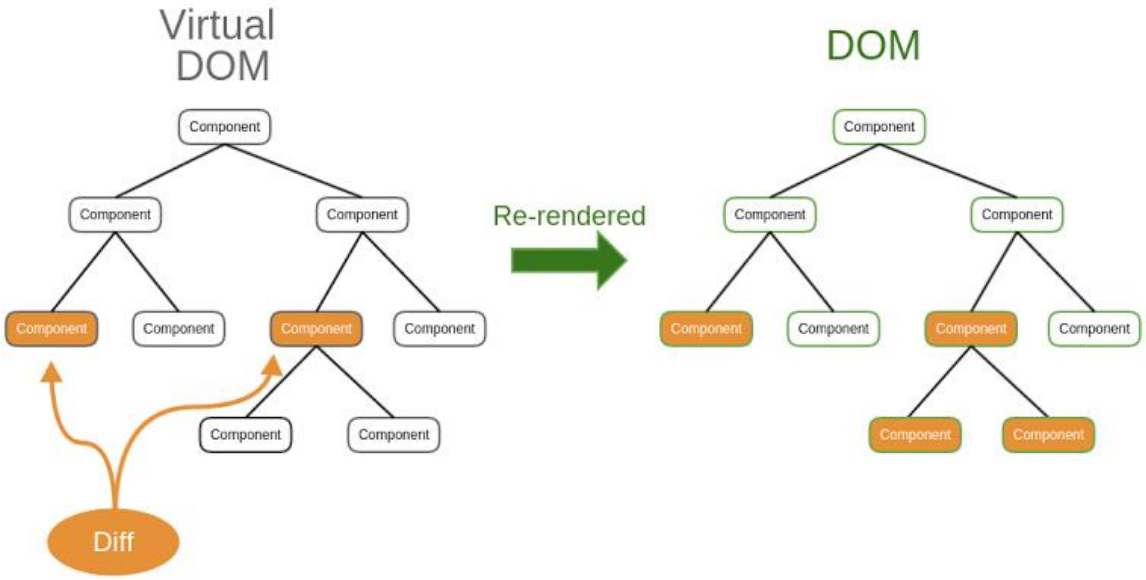
Για την υλοποίηση της πτυχιακής εργασίας χρησιμοποιήθηκαν τα functional Components λόγω ότι η React προτείνει την χρήση τους.

Μαζί με τα functional Components, παρουσίασαν και τα hooks, που επιτρέπουν την χρήση state και side effects απομονώνοντας τα στο κάθε component. Η React προσφέρει κάποια in-build hooks τα οποία είναι πολύ βασικά:

- useState: το πιο βασικό hook, επιτρέπει στο functional component να διατηρεί και να ενημερώνει states. Επιστρέφει μία λίστα με δύο elements: current state και μία μέθοδο για να κάνει ενημέρωση το state.
- useEffect: επιτρέπει τα functional components να εκτελούν side effects. Αντικαθιστά τις μεθόδους κύκλου ζωής όπως componentDidMount, componentDidUpdate και componentWillUnmount που υπάρχουν σε class based components. Εκτελείται μετά από κάθε render ή ανάλογα τι περιέχει η λίστα εξάρτησης.
- useContext: παρέχει έναν τρόπο να καταναλώνουν τιμές από το context API της React. Δέχεται ένα αντικείμενο context που δημιουργείται με το React.createContext και επιστρέφει την τρέχουσα τιμή context. Αυτό το hook είναι χρήσιμο για την πρόσβαση σε γενικά δεδομένα ή ρυθμίσεις θέματος (theme) σε όλο το δέντρο των components χωρίς τον προσδιορισμό τους μέσω των props.
- useRef Components: επιστρέφει ένα mutable αντικείμενο αναφοράς (ref) που διατηρείται για ολόκληρη τη διάρκεια ζωής του component. Χρησιμοποιείται συνήθως για την πρόσβαση σε στοιχεία DOM ή για τη διατήρηση τιμών μεταξύ remder χωρίς να προκαλεί re-renders στα components. Αντίθετα με το χαρακτηριστικό ref στα class based components, το useRef μπορεί να κρατήσει μια μεταβλητή τιμή που διατηρείται ανάμεσα στα renders χωρίς να προκαλεί re-render όταν η τιμή της αλλάζει.
- useMemo Components: χρησιμοποιείται για τη μνημονική καταχώρηση και επανεκτέλεση μεθόδων και προσπελάσεων εξαρτημένων από δεδομένα. Χρησιμοποιείται για την αποφυγή υπολογισμών function κατά τα re-renders, αν το input δεδομένων δεν έχει αλλάξει.
- useCallback Components: χρησιμοποιείται για τη μνημονική δημιουργία μιας callback μεθόδου, που μπορεί να διατηρηθεί μεταξύ των re-renders. Χρησιμοποιείται για την αποφυγή της δημιουργίας νέας function κατά τα re-renders, αν τα εξαρτώμενα δεδομένα δεν έχουν αλλάξει.

Μία ακόμη αλλαγή που έφερε η React στον κόσμο της JavaScript ήταν το Virtual DOM. Πρώτα όμως για να καταλάβουμε τι προσφέρει θα πρέπει να μιλήσουμε για το τι είναι το DOM. Η HTML είναι η γλώσσα των ιστοσελίδων. Παρέχει τη δομή της ιστοσελίδας με πολλές ειδικές ετικέτες (tags), συμπεριλαμβανομένου του τρόπου σύνδεσης πολλαπλών σελίδων μαζί. Η δομή μιας ιστοσελίδας αναπαρίσταται ως έγγραφο μοντέλων αντικειμένων ως δέντρο. Η γλώσσα προγραμματισμού JavaScript μπορεί να αλλάξει τη δομή αυτού του αντικειμένου εγγράφου για να προσδώσει δυναμική συμπεριφορά στις ιστοσελίδες. Το DOM αποτελεί την προγραμματιστική διεπαφή για τα έγγραφα ιστοσελίδων με δομή δέντρου. Το δέντρο έγγραφου ονομάζεται δέντρο DOM. Οι συχνές ενημερώσεις στο DOM είναι δαπανηρές. Μπορεί να επιβραδύνουν την απόδοση της ιστοσελίδας και να την καθιστούν αργή. Καθώς το DOM αναπαρίσταται με μια δομή δέντρου, η ανάκτηση και η ενημέρωση είναι συνήθως ταχύτερες από την αναπαραγωγή. Ωστόσο, μπορεί επίσης να είναι δαπανηρές εάν

πρέπει να διατρέξουμε ένα μεγάλο τμήμα του δέντρου DOM για να βρούμε τον κόμβο προς ενημέρωση. Η React δεν ενημερώνει ποτέ απευθείας τον αρχικό DOM (εκτός αν μια περίπτωση χρήσης του προγραμματιστή το απαιτεί). Για κάθε αντικείμενο DOM, δημιουργείται μια αντίστοιχη αντίγραφη στη μνήμη. Αυτό το αντίγραφο ονομάζεται Virtual DOM. Στο Virtual DOM δέντρο, κάθε στοιχείο αντιπροσωπεύεται από έναν κόμβο (node). Ένα νέο Virtual DOM δέντρο θα δημιουργηθεί κάθε φορά που αλλάζει το state του στοιχείου. Ο αλγόριθμος διαφοροποίησης (diffing) της React συγκρίνει το τρέχον Virtual DOM δέντρο με την προηγούμενή του έκδοση. Τέλος, το Virtual DOM χρησιμοποιεί τον αλγόριθμο για να ενημερώσει τον πραγματικό DOM με τις διαφορές. Στην Εικόνα 3.2 *Virtual DOM re-render process*, δείχνει ένα παράδειγμα για το πως γίνεται το re-render στην React.



Εικόνα 3.2: Virtual DOM re-render process

Τα Synthetic Events είναι πολύ παρόμοια με τα native events, ωστόσο, με τα Synthetic Events, το ίδιο API εφαρμόζεται σε πολλούς περιηγητές. Καλύπτουν το native event του περιηγητή μέσω του nativeEvent και παρέχουν μια ομοιόμορφη διεπαφή και συνεπή/ίδια συμπεριφορά στο πιο πάνω επίπεδο.

Η React προσφέρει τέσσερις διαφορετικές προσεγγίσεις για τη δημιουργία και την παράδοση διαδραστικών εφαρμογών:

- SSR (Server-Side Rendering): η αρχική αναπαραγωγή της εφαρμογής React δημιουργείται στην πλευρά του διακομιστή (server) και αποστέλλεται στον περιηγητή του χρήστη ως έτοιμο HTML. Αυτό επιτρέπει στο χρήστη να βλέπει γρήγορα το περιεχόμενο της εφαρμογής και είναι χρήσιμο για τη βελτίωση του SEO και τη βελτίωση του χρόνου απόκρισης της σελίδας.
- CSR (Client-Side Rendering): αναπαρίσταται εξ' ολοκλήρου στον περιηγητή του χρήστη, με τη χρήση JavaScript, χωρίς να υπάρχει προκατασκευασμένο HTML από τον διακομιστή. Αυτό επιτρέπει τη δημιουργία πλούσιων διαδραστικών εφαρμογών, αλλά μπορεί να οδηγήσει σε αργότερους χρόνους αναμονής μέχρι την πρώτη αναπαραγωγή του περιεχομένου.
- SSG (Static Site Generation): δημιουργείται κατά τον χρόνο της κατασκευής και αποθηκεύεται ως στατικό HTML αρχείο, το οποίο στη συνέχεια αποστέλλεται στον περιηγητή

του χρήστη. Αυτό προσφέρει ταχύτερους χρόνους φόρτωσης και μικρότερους χρόνους απόκρισης, καθώς το περιεχόμενο είναι ήδη προκατασκευασμένο.

- ISR (Incremental Static Regeneration): είναι μια παραλλαγή της SSG, όπου τα στατικά HTML αρχεία δημιουργούνται δυναμικά κατά την εκτέλεση, ανανεώνονται και αναπροσδιορίζονται όταν υπάρχουν αλλαγές στο περιεχόμενο της εφαρμογής. Αυτό επιτρέπει την απόδοση στατικών σελίδων με δυνατότητα δυναμικής ενημέρωσης, προσφέροντας την καλύτερη απόδοση και την πιο ενημερωμένη προβολή του περιεχομένου.

3.2.2 React Router

Η React είναι μία βιβλιοθήκη η οποία εστιάζει στην παροχή ενός απλού και αποτελεσματικού τρόπου διαχείρισης του UI των εφαρμογών ιστού. Για αυτό χρειάστηκε η React Router για να μια εφαρμογή React με πλοήγηση σε πολλές σελίδες [24]. Είναι μία βιβλιοθήκη που μας επιτρέπει να διαχειριστούμε την πλοήγηση σε πλευρά client και σε πλευρά server. Επιτρέπει τη δημιουργία μονοσέλιδων ιστοσελίδων ή εφαρμογών για κινητές συσκευές σε React Native εφαρμογές που επιτρέπουν την πλοήγηση χωρίς ανανέωση της σελίδας. Επιπλέον, μας επιτρέπει να χρησιμοποιούμε τις λειτουργίες ιστορικού του περιηγητή διατηρώντας τη σωστή προβολή της εφαρμογής.

3.2.3 Chart.js 2

Η βιβλιοθήκη Chart.js 2 για την React παρέχει έναν εύκολο και ισχυρό τρόπο για τη δημιουργία διαγραμμάτων και γραφημάτων στις εφαρμογές React. Με τη χρήση αυτής της βιβλιοθήκης, οι προγραμματιστές μπορούν να προσθέσουν εύκολα διαγράμματα στις εφαρμογές τους για να απεικονίσουν δεδομένα και να παρέχουν εντυπωσιακές αναλύσεις. Βασίζεται στη δημοφιλή βιβλιοθήκη Chart.js και παρέχει μια React-friendly προσέγγιση για τη χρήση των διαγραμμάτων σε εφαρμογές React. Με την ενσωμάτωση της Chart.js 2 οι προγραμματιστές μπορούν να εκμεταλλευτούν τη δυνατότητα της React να διαχειρίζεται την αλληλεπίδραση με το DOM και του state management, ενώ ταυτόχρονα απολαμβάνουν την ευκολία και την ευελιξία που προσφέρει η Chart.js για τη δημιουργία εντυπωσιακών διαγραμμάτων [25].

Το Chart.js ξεκινά το 2013, όταν ο Nick Downie, ένας προγραμματιστής και σχεδιαστής λογισμικού, δημιούργησε τη βιβλιοθήκη με σκοπό τη δημιουργία ελαφρών, εύκολων στη χρήση και ευέλικτων διαγραμμάτων για τις ιστοσελίδες. Η βιβλιοθήκη παρέχει έναν απλό API που επιτρέπει στους προγραμματιστές να δημιουργούν διαγράμματα με λίγες γραμμές κώδικα.

Προσφέρει ένα ευρύ φάσμα τύπων διαγραμμάτων και γραφημάτων, καθιστώντας την ιδανική λύση για την αναπαράσταση δεδομένων σε διάφορες περιπτώσεις. Μεταξύ των διαγραμμάτων που υποστηρίζει περιλαμβάνονται οι γραφικές παραστάσεις, τα διαγράμματα γραμμών, τα διαγράμματα πίτας, τα διαγράμματα δοχείων, τα διαγράμματα ραντάρ, τα διαγράμματα διαστημάτων και πολλά άλλα. Αυτό επιτρέπει στους προγραμματιστές να επιλέξουν τον κατάλληλο τύπο διαγράμματος ανάλογα με τον τύπο και την πολυπλοκότητα των δεδομένων που θέλουν να αναπαραστήσουν. Μπορεί επίσης να δημιουργήσει διαδραστικά διαγράμματα που επιτρέπουν στους χρήστες να αλληλεπιδρούν με τα δεδομένα. Αυτό δίνει τη δυνατότητα για πιο εμπλουτισμένες και δυναμικές εμπειρίες χρήστη.

Ένα άλλο σημαντικό χαρακτηριστικό του Chart.js είναι η δυνατότητα ανταπόκρισης (responsive design). Η βιβλιοθήκη προσφέρει ενσωματωμένη υποστήριξη για τη δημιουργία διαγραμμάτων που προσαρμόζονται αυτόματα σε διάφορες διαστάσεις οθονών και συσκευών. Αυτό σημαίνει ότι τα διαγράμματα που δημιουργούνται με το Chart.js μπορούν να προβάλλονται με επιτυχία σε desktop

υπολογιστές, φορητές συσκευές, κινητά τηλέφωνα και άλλες συσκευές, προσφέροντας μια συνεπή και επαγγελματική εμπειρία στους χρήστες.

Αν και το Chart.js είναι εύκολο στη χρήση, η διαδικασία δημιουργίας πολύπλοκων διαγραμμάτων μπορεί να γίνει πιο περίπλοκη και χρονοβόρα. Η ανάπτυξη προηγμένων διαγραμμάτων που απαιτούν πολλές προσαρμογές και διαμόρφωση μπορεί να απαιτήσει επιπλέον προσπάθεια και κατάρτιση. Η υποστήριξη και η τεκμηρίωση του Chart.js μπορεί να μην είναι τόσο εκτεταμένες όσο θα θέλαν ορισμένοι χρήστες. Ενώ υπάρχει μια ενεργή κοινότητα που συνεισφέρει στη βελτίωση της βιβλιοθήκης, η ποιότητα της υποστήριξης και της τεκμηρίωσης μπορεί να διαφέρει ανάλογα με την περίπτωση. Αυτό μπορεί να οδηγήσει σε δυσκολίες για τους χρήστες που χρειάζονται βοήθεια ή έχουν ερωτήσεις για τη χρήση ή την αντιμετώπιση προβλημάτων με το Chart.js.

Τέλος, το documentation του Chart.js είναι συνεχώς ενημερωμένο και περιλαμβάνει λεπτομερείς οδηγίες με παραδείγματα κώδικα, επεξηγήσεις παραμέτρων και πληροφορίες σχετικά με τις διάφορε δυνατότητες και λειτουργίες της βιβλιοθήκης. Διαθέτει συνήθως την πιο πρόσφατη πληροφορία για τις νέες εκδόσεις και τις αλλαγές στη βιβλιοθήκη. Αυτό εξασφαλίζει ότι οι χρήστες έχουν πρόσβαση σε ενημερωμένες οδηγίες και πληροφορίες, βοηθώντας τους να διατηρούν τις εφαρμογές τους ενημερωμένες και λειτουργικές [26].

3.2.4 Redux toolkit

Η Redux Toolkit αποτελεί μια εξέλιξη της δημοφιλούς βιβλιοθήκης διαχείρισης κατάστασης για εφαρμογές React, της Redux. Η Redux είναι μια βιβλιοθήκη που έχει αποκτήσει μεγάλη αναγνωρισιμότητα στην κοινότητα του React για τη δυνατότητά της να διαχειρίζεται αποτελεσματικά την κατάσταση της εφαρμογής, επιτρέποντας την ανάπτυξη προβλέψιμων και εύκολα διαχειρίσιμων εφαρμογών.

Η Redux αρχικά παρουσιάστηκε από τον Dan Abramov και τον Andrew Clark το 2015 και απέκτησε γρήγορα μεγάλη απήχηση στην κοινότητα της React. Αυτό οφείλεται στην απλότητα και τη σαφήνεια της σχεδίασής της, καθώς και στο γεγονός ότι παρέχει μια ολοκληρωτική λύση για τη διαχείριση της κατάστασης σε οποιοδήποτε μέγεθος εφαρμογής React.

Ωστόσο, με την πάροδο του χρόνου, ορισμένοι προγραμματιστές είχαν την εντύπωση ότι η Redux έβαζε πολλά επιπλέον βάρη και συντακτικές προϋποθέσεις στη διαδικασία ανάπτυξης. Αυτό οφείλεται κυρίως στην ανάγκη για πολλούς κώδικες boilerplate για τη διαχείριση των actions, reducers και του store.

Εδώ είναι που έρχεται στο προσκήνιο η Redux Toolkit. Η Redux Toolkit είναι μια επίσημη πρόταση από τον ίδιο τον δημιουργό της Redux, Dan Abramov, η οποία αποσκοπεί στην απλοποίηση της διαδικασίας χρήσης της Redux και τη μείωση του boilerplate κώδικα.

Το createSlice αποτελεί ένα από τα βασικά χαρακτηριστικά της Redux Toolkit που συμβάλλει στην απλοποίηση της διαδικασίας δημιουργίας reducers και actions. Αυτή η μέθοδος επιτρέπει στους προγραμματιστές να δημιουργούν reducers και τα σχετικά actions με λιγότερο κώδικα και μεγαλύτερη ευκολία, παρέχοντας ένα πιο δομημένο και συντηρήσιμο σύνολο λειτουργιών.

Το createSlice λειτουργεί ως ένα είδος "εργαστήριο" για τη δημιουργία των reducers και των αντίστοιχων actions. Με το createSlice, οι προγραμματιστές ορίζουν το σύνολο των actions και των reducers που σχετίζονται με ένα συγκεκριμένο τμήμα του status της εφαρμογής. Κατά τη διαδικασία δημιουργίας, το createSlice δέχεται ένα αρχικό state, ένα ή περισσότερα reducers και ένα όνομα πρόθεμα που χρησιμοποιείται για τη δημιουργία των actions.

Ένα από τα σημαντικά πλεονεκτήματα του createSlice είναι ότι απλοποιεί τη σύνταξη του κώδικα, επιτρέποντας στους προγραμματιστές να ορίζουν reducers και actions με λιγότερες γραμμές κώδικα. Επίσης, η δομή του createSlice παρέχει μια οργανωμένη προσέγγιση για τη διαχείριση του κώδικα, καθώς οι σχετικές λειτουργίες βρίσκονται συγκεντρωμένες σε ένα μέρος.

Επιπλέον, το createSlice αποτελεί έναν πολύ ευέλικτο τρόπο για τη δημιουργία reducers και actions, καθώς επιτρέπει στους προγραμματιστές να ορίζουν πολλαπλά reducers για διάφορες περιπτώσεις χρήσης και να τα οργανώνουν με λογικό τρόπο.

Ένα ακόμη ισχυρό εργαλείο που παρέχει είναι το createApi, για τη διαχείριση αιτημάτων δικτύου. Επιτρέπει στους προγραμματιστές να ορίσουν έναν αριθμό από endpoints, κάθε ένα από τα οποία αντιστοιχεί σε μια συγκεκριμένη διαδρομή στο API. Κάθε endpoint παρέχει μια σειρά από reducers που διαχειρίζονται την κατάσταση των δεδομένων που προκύπτουν από το αίτημα. Η δυνατότητά του να προσφέρει ενσωματωμένη υποστήριξη για διάφορες μεθόδους HTTP, όπως GET, POST, PUT και DELETE είναι ένα από τα πλεονεκτήματα του. Αυτό επιτρέπει στους προγραμματιστές να αλληλοεπιδρούν με τον διακομιστή API με ευελιξία και ευκολία. Όπως με το createSlice, απλοποιεί την περιπλοκότητα που έχει η διαχείριση των δικτυακών αιτημάτων και είναι οργανωμένα σε ένα ενιαίο μέρος [27].

3.2.5 MUI (Material-UI)

Η βιβλιοθήκη MUI (Material-UI) για την React ξεκίνησε ως μια πρωτοβουλία από έναν προγραμματιστή με το όνομα Olivier Tassinari. Κυκλοφόρησε για πρώτη φορά το 2014, με στόχο να φέρει τις αρχές του Material Design της Google σε εφαρμογές της React. Ο Tassinari εμπνεύστηκε από τις οδηγίες σχεδίασης του Material Design της Google, οι οποίες πρόσφεραν μια συνεκτική γλώσσα σχεδίασης για τη δημιουργία διεπαφών που είναι ελκυστικές οπτικά και φιλικές προς τον χρήστη σε διάφορες πλατφόρμες.

Αρχικά, το Material-UI ξεκίνησε ως ένα μικρό έργο, αλλά απέκτησε γρήγορα δημοτικότητα μέσα στην κοινότητα της React λόγω της απλότητάς του, της ευελιξίας του και της συμμόρφωσής του στις αρχές του Material Design. Καθώς το έργο απέκτησε ώθηση, περισσότεροι προγραμματιστές συνέβαλαν στην ανάπτυξή του, επεκτείνοντας τις λειτουργίες και τις δυνατότητές του.

Η βιβλιοθήκη παρείχε components που υλοποιούσαν τις προδιαγραφές του Material Design, επιτρέποντας στους προγραμματιστές να ενσωματώσουν εύκολα αυτά τα στοιχεία στις εφαρμογές της React. Αυτό έκανε ευκολότερη τη δημιουργία διεπαφών που ακολουθούν τις οδηγίες του Material Design χωρίς να χρειάζεται να χτίσουν τα πάντα από την αρχή.

Το 2021, πήραν την απόφαση να αλλάξει το όνομα από Material-UI σε MUI. Κινήθηκε κυρίως από την επιθυμία να απλοποιηθεί η επωνυμία, να δηλωθεί η ανεξαρτησία από οποιοδήποτε συγκεκριμένο σύστημα σχεδιασμού όπως το Material Design και να ελκυσθεί ένας ευρύτερος κύκλος προγραμματιστών και σχεδιαστών. Το MUI λειτουργεί ως ένα πιο σύντομο, πιο καθολικό όνομα που απλοποιεί την επικοινωνία σχετικά με τη βιβλιοθήκη και αντικατοπτρίζει τον ευρύτερο στόχο της παροχής ενός ευέλικτου εργαλείου περιβάλλοντος χρήστη για εφαρμογές React [28].

Εντός του οικοσυστήματος του Material-UI, υπάρχουν τρία κύρια πακέτα: MUI Core, MUI X και Material-UI Icons. Ας εξετάσουμε κάθε ένα από αυτά:

- MUI Core: είναι το θεμελιώδες πακέτο του οικοσυστήματος MUI. Παρέχει μια εκτεταμένη συλλογή προ-σχεδιασμένων στοιχείων React που ακολουθούν τις αρχές της Material Design. Αυτά τα στοιχεία περιλαμβάνουν κουμπιά, εισόδους, διαλόγους, πλέγματα, στοιχεία

πλοήγησης και πολλά άλλα, προσφέροντας στους προγραμματιστές έναν συνεπή και προσαρμόσιμο τρόπο δημιουργίας όμορφων διεπαφών χρήστη. Το MUI Core μπορεί να διαμορφωθεί πολύ, επιτρέποντας στους προγραμματιστές να επιλέξουν και να χρησιμοποιήσουν τα στοιχεία που χρειάζονται για τα έργα τους. Επίσης, προσφέρει εκτενείς επιλογές προσαρμογής μέσω θεμάτων, επιτρέποντας στους προγραμματιστές να προσαρμόσουν εύκολα το οπτικό στυλ των εφαρμογών τους για να ταιριάζουν με το brand ή τις απαιτήσεις σχεδίασης τους. Το πακέτο αποτελεί τη βάση του Material-UI και είναι απαραίτητο για τη δημιουργία εφαρμογών React με αισθητική του Material Design.

- MUI X: προσφέρει μια συλλογή προηγμένων React components. Αυτά τα στοιχεία σχεδιάστηκαν για τη δημιουργία πολύπλοκων και πλούσιων σε δεδομένα εφαρμογών. Μερικά παραδείγματα περιλαμβάνουν το Data Grid, τα Date and Time Pickers, τα Διαγράμματα και άλλα. Το MUI X προσφέρει μερικές λειτουργίες δωρεάν αλλά τα πιο προηγμένα είναι επί πληρωμής.
- Material-UI Icons: είναι ένα ξεχωριστό πακέτο που παρέχει έναν πλήρη σύνολο εικονιδίων που ακολουθούν τις οδηγίες της Google σχετικά με την εικονογραφία της Material Design. Αυτά τα εικονίδια χρησιμοποιούνται συνήθως σε συνδυασμό με άλλα στοιχεία του Material-UI για να βελτιώσουν την οπτική αναπαράσταση και τη χρηστικότητα της διεπαφής χρήστη. Παρέχει μια ευρεία γκάμα εικονιδίων που καλύπτουν διάφορες κατηγορίες, όπως δράση, ειδοποίηση, επικοινωνία, περιεχόμενο και άλλα.

3.3 Backend

3.3.1 Node.js

Η Node.js είναι ένα ανοιχτού κώδικα, περιβάλλον εκτέλεσης JavaScript στην πλευρά του server, που έχει χτιστεί στον μηχανισμό JavaScript V8 του Chrome. Επιτρέπει στους προγραμματιστές να εκτελούν κώδικα JavaScript εκτός περιβάλλοντος περιήγησης (browser), κάνοντας δυνατή τη δημιουργία κλιμακούμενων και υψηλής απόδοσης διαδικτυακών εφαρμογών. Η Node.js χρησιμοποιεί ένα μη-αποκλειστικού μοντέλο I/O και event-driven που την καθιστά ελαφριά και αποτελεσματική, καθιστώντας την ιδανική για πραγματικού χρόνου εφαρμογές που χρειάζεται να διαχειριστεί ένα μεγάλο αριθμό συνδέσεων ταυτόχρονα.

Η ιστορία της Node.js ξεκινάει το 2009 όταν ο Ryan Dahl, ένας μηχανικός λογισμικού, που παρουσίασε το έργο στο European JSConf. Ο Dahl αναζητούσε έναν τρόπο για να δημιουργήσει κλιμακούμενα δικτυακά προγράμματα και φανταζόταν ένα περιβάλλον εκτέλεσης που θα μπορούσε να αξιοποιήσει το event-driven της JavaScript για την ανάπτυξη στην πλευρά του server. Σκοπός του ήταν να αντιμετωπίσει τα περιορισμένα στοιχεία των παραδοσιακών τεχνολογιών στην πλευρά του server, τα οποία συχνά αντιμετώπιζαν προβλήματα με την ταυτόχρονη χρήση και bottlenecks.

Η Node.js έλαβε πρώιμη αναγνώριση λόγω της καινοτόμου προσέγγισής της και της αυξανόμενης δημοτικότητας της JavaScript. Το μοντέλο μη-αποκλειστικού I/O της επέτρεψε στους προγραμματιστές να διαχειρίζονται πολλές συνδέσεις ταυτόχρονα χωρίς να αποκλείεται η εκτέλεση άλλων λειτουργιών, οδηγώντας σε σημαντικές βελτιώσεις στην απόδοση.

Το 2010, η Node.js έλαβε σημαντική ώθηση όταν η Joyent, μια εταιρεία cloud computing, χορήγησε επίσημη υποστήριξη στο έργο και παρείχε πόρους για την ανάπτυξή της. Με την υποστήριξη της Joyent, η Node.js είδε ταχεία ανάπτυξη και υιοθέτηση εντός της κοινότητας προγραμματιστών.

Single-Threaded, Event Loop: Η Node.js λειτουργεί σε ένα Single-Threaded μοντέλο σάρωσης γεγονότων, το οποίο διαχειρίζεται αποτελεσματικά τις λειτουργίες I/O και τις επανακλήσεις γεγονότων. Η σάρωση γεγονότων ελέγχει συνεχώς την ουρά γεγονότων για εκκρεμή γεγονότα και τα επεξεργάζεται με non-blocking τρόπο. Ενώ η ίδια η σάρωση γεγονότων είναι Single-Threaded, η Node.js μπορεί ακόμα να επιτύχει την συνυφασμένη διακριτικότητα μέσω ασύγχρονων λειτουργιών, επιτρέποντας της να χειρίζεται πολλαπλά αιτήματα ταυτόχρονα χωρίς τη δημιουργία επιπλέον threads. Αυτό το ελαφρύ μοντέλο διακριτικότητας μειώνει τον πρόσθετο φόρτο πόρων και απλοποιεί την ανάπτυξη εφαρμογών.

Cross-Platform Compatibility: σχεδιάστηκε για να τρέχει σε πολλαπλές πλατφόρμες, συμπεριλαμβανομένων των Windows, macOS και διάφορων λειτουργικών συστημάτων Unix. Η δυνατότητά της για συμβατότητα πολλαπλών πλατφορμών τη καθιστά μια ευέλικτη επιλογή για την ανάπτυξη εφαρμογών που χρειάζονται να τρέχουν σε διαφορετικά περιβάλλοντα χωρίς σημαντικές τροποποιήσεις. Αυτή η ευελιξία επιτρέπει στους προγραμματιστές να γράφουν κώδικα μία φορά και να τον αναπτύσσουν σε διάφορες πλατφόρμες, απλοποιώντας τη διαδικασία ανάπτυξης.

Real-Time Capabilities: στη δημιουργία εφαρμογών πραγματικού χρόνου που απαιτούν διπλή κατεύθυνση επικοινωνίας μεταξύ πελατών και διακομιστών, όπως εφαρμογές συνομιλίας, online πλατφόρμες παιχνιδιών και εργαλείων συνεργασίας, η Node.js ξεχωρίζει. Η φύση που λειτουργεί με βάση τα γεγονότα και η υποστήριξη για το WebSocket επιτρέπουν στους προγραμματιστές να υλοποιήσουν εύκολα πρωτόκολλα επικοινωνίας πραγματικού χρόνου. Με frameworks όπως το Socket.IO, οι προγραμματιστές μπορούν να δημιουργήσουν διαδραστικές εφαρμογές που προσφέρουν πραγματικού χρόνου εμπειρίες στους χρήστες.

Ένα από τα βασικά χαρακτηριστικά που συνέβαλαν στην επιτυχία της Node.js είναι ο διαχειριστής πακέτων του, το npm (Node Package Manager). Το npm επιτρέπει στους προγραμματιστές να εγκαθιστούν, να διαχειρίζονται και να μοιράζονται εύκολα επαναχρησιμοποιήσιμα πακέτα κώδικα, γνωστά ως modules, που επιταχύνουν σημαντικά τη διαδικασία ανάπτυξης [29].

Dependency Management: Το npm απλοποιεί τη διαχείριση των dependency για τις εφαρμογές Node.js παρέχοντας ένα κεντρικό κατάλογο πακέτων και ένα απλό μηχανισμό για την καθορισμό των dependency στο αρχείο package.json του έργου. Οι προγραμματιστές μπορούν να δηλώσουν τα απαιτούμενα dependency για τα έργα τους, συμπεριλαμβανομένων συγκεκριμένων εκδόσεων ή εύρων, και το npm χειρίζεται αυτόματα την εγκατάσταση και την επίλυση των εξαρτήσεων. Αυτή η διαδικασία εξυπηρετεί τη συνέπεια και την αναπαραγωγιμότητα σε διαφορετικά περιβάλλοντα και απλοποιεί τη διαχείριση των εξαρτήσεων του έργου.

Package Publishing and Sharing: Το npm επιτρέπει στους προγραμματιστές να δημοσιεύουν τα δικά τους πακέτα στο npm registry, καθιστώντας τα προσβάσιμα στην ευρύτερη κοινότητα. Αυτό ενθαρρύνει τη συνεργασία και την κοινοποίηση γνώσεων μεταξύ των προγραμματιστών, καθώς μπορούν να συνεισφέρουν με τα δικά τους modules και να επωφεληθούν από τις συνεισφορές άλλων. Το npm registry λειτουργεί ως ένα δυναμικό οικοσύστημα όπου οι προγραμματιστές μπορούν να ανακαλύπτουν, να μοιράζονται και να συνεργάζονται σε ανοικτές πηγές έργα, προωθώντας την καινοτομία και την πρόοδο στην κοινότητα της Node.js.

Scripts and Automation: επίσης το npm περιλαμβάνει υποστήριξη για την εκτέλεση σεναρίων και εργασιών αυτοματισμού μέσω του αρχείου package.json. Οι προγραμματιστές μπορούν να ορίζουν προσαρμοσμένα σενάρια για την εκτέλεση κοινών εργασιών, όπως κατασκευή, δοκιμή, ελέγχους σύνταξης και αναπτυξιακές διαδικασίες, βελτιώνοντας την ροή εργασίας ανάπτυξης. Τα σενάρια του

npm μπορούν να εκτελεστούν χρησιμοποιώντας απλές εντολές όπως npm run «όνομα-σεναρίου», παρέχοντας μια βολική μέθοδο αυτοματισμού επαναλαμβανόμενων εργασιών και βελτίωσης της παραγωγικότητας.

Μέσα στα χρόνια, η Node.js έχει εξελιχθεί σε ένα ώριμο περιβάλλον, που τροφοδοτεί μια ευρεία γκάμα εφαρμογών, από διακομιστές και APIs μέχρι εφαρμογές γραφικής διασύνδεσης χρήστη (GUI) και συσκευές του διαδίκτυο των πραγμάτων (IoT). Η ευελιξία της, η απόδοσή της και το εκτεταμένο οικοσύστημα των πακέτων το έχουν καταστήσει μια δημοφιλή επιλογή για τη δημιουργία σύγχρονων, κλιμακούμενων και πραγματικού χρόνου εφαρμογών.

Η Node.js συνεχίζει να εξελίσσεται, με τακτικές ενημερώσεις και νέα χαρακτηριστικά που εισάγονται για να ενισχύσουν τις δυνατότητές της και να αντιμετωπίσουν τις εξελισσόμενες ανάγκες των προγραμματιστών [30].

3.3.2 Express

Η Express είναι ένα ισχυρό και ευέλικτο framework για την Node.js, σχεδιασμένο για να απλοποιήσει τη διαδικασία της δημιουργίας εφαρμογών και API στο web. Παρέχει ένα ελάχιστο, αλλά ισχυρό σύνολο χαρακτηριστικών για τη δημιουργία εφαρμογών στην πλευρά του server, την χειριστικότητα των αιτήσεων και αποκρίσεων HTTP, το routing, την υποστήριξη middleware και πολλά άλλα. Το Express χρησιμοποιείται ευρέως στην κοινότητα του Node.js και γνωρίζει αναγνώριση για την απλότητα, την κλιμακωσιμότητα και την απόδοσή του.

Η ιστορία της Express χρονολογείται από το 2010, όταν ο TJ Holowaychuk την δημιούργησε ως ένα έργο για να παρέχει έναν πιο ελαφρύ και ανεπηρέαστο εναλλακτικό λογισμικό στα υπάρχοντα frameworks στον κόσμο της Node.js. Ο Holowaychuk είχε σκοπό να αναπτύξει ένα framework που θα δίνει προτεραιότητα στην απλότητα, την ευελιξία και την ευκολία χρήσης, παρέχοντας ταυτόχρονα μια στέρεη βάση για τη δημιουργία εφαρμογών στο web. Κυκλοφόρησε η Express ως open-source υπό την άδεια MIT, επιτρέποντας στους προγραμματιστές να συνεισφέρουν στην ανάπτυξή του και να το προσαρμόσουν στις ανάγκες τους.

Η Express είναι ένα ισχυρό και ευέλικτο framwork για την Node.js που παρέχει μια ευρεία γκάμα εργαλείων και χαρακτηριστικών για τη δημιουργία RESTful APIs. Παρακάτω θα αναλυθούν μερικά από τα κύρια εργαλεία και συστατικά που προσφέρει η Express για τη δημιουργία APIs.

Routing: παρέχει ένα ανθεκτικό σύστημα δρομολόγησης που επιτρέπει να ορίζονται σημεία πρόσβασης (endpoints) στο API και να αντιστοιχούνε σε αντίστοιχες συναρτήσεις (functions). Μπορεί να χρησιμοποιηθούν μεθόδους HTTP όπως GET, POST, PUT και DELETE για να ορίσετε διαδρομές (routes) για την ανάκτηση, δημιουργία, ενημέρωση και διαγραφή πόρων. Το σύστημα δρομολόγησης της Express υποστηρίζει τόσο στατικές όσο και δυναμικές διαδρομές με παραμέτρους, κάτι που καθιστά εύκολη τη δημιουργία ευέλικτων APIs.

Middleware: Οι λειτουργίες μεσολάβησης είναι ένα κύριο χαρακτηριστικό της Express που επιτρέπει να προστεθεί επιπλέον λειτουργικότητα στον κύκλο αιτήσεων-αποκρίσεων (request-response) του API. Οι middlewares της Express μπορούν να εκτελούν εργασίες όπως η ανάλυση των σωμάτων αιτήσεων, η χειρισμός του CORS (Cross-Origin Resource Sharing), η πιστοποίηση των χρηστών, η καταγραφή των αιτήσεων, η συμπίεση των αποκρίσεων και πολλά άλλα. Μπορεί να γίνει χρήση ενσωματωμένων middlewares που παρέχονται από την Express ή να γραφτούν προσαρμοσμένους middlewares για να ικανοποιήσει τις συγκεκριμένες απαιτήσεις που μπορεί ένας προγραμματιστής να θελήσει.

Body Parsing: η Express περιλαμβάνει middlwares για την ανάλυση των σωμάτων αιτήσεων (request bodies) σε διάφορες μορφές, συμπεριλαμβανομένων JSON, URL-encoded και multipart forms. Αυτό καθιστά εύκολη την αντιμετώπιση των εισερχόμενων δεδομένων που αποστέλλονται από τους πελάτες και την εξαγωγή σχετικών πληροφοριών από τα σώματα αιτήσεων (request bodies). Υπάρχει η δυνατότητα να χρησιμοποιηθούν middlwares όπως τα express.json() και express.urlencoded() για να αναλυθούν αυτόματα τα σώματα αιτήσεων (request bodies) και να κάνει τα δεδομένα διαθέσιμα στους χειριστές διαδρομών.

Error Handling: ενσωματωμένους middlewares για την αντιμετώπιση σφαλμάτων που προκύπτουν κατά την επεξεργασία των αιτήσεων. Είναι εφικτό να χρησιμοποιηθεί την συνάρτηση next για να περάσουν τα σφάλματα στον error handling middleware της Express, ο οποίος θα τα αιχμαλωτίσει και θα τα διαχειριστεί. Ο error handling middleware μπορεί να δημιουργήσει κατάλληλες αποκρίσεις σφαλμάτων, να καταγράφει τα σφάλματα και να εκτελεί εργασίες καθαρισμού όπως απαιτείται, εξασφαλίζοντας ότι το API σας παραμένει ανθεκτικό και αξιόπιστο.

Routing Middleware: η Express επιτρέπει την οργάνωση των middlewares και τους χειριστές δρομολόγησης σε μονάδες που ονομάζονται δρομολογητές (routers). Οι δρομολογητές (routers) επιτρέπουν να ομαδοποιηθούν σχετικές διαδρομές και middlewares μαζί, κάνοντας τον κώδικά πιο οργανωμένο και ευκολότερο στη συντήρηση. Επιτρέπει την δημιουργία πολλαπλών δρομολογητών για διαφορετικά μέρη του API και να τοποθετούνται σε συγκεκριμένες διαδρομές στην εφαρμογή, επιτρέποντας την modular και κλιμακούμενη σχεδίαση του API.

Template Engines: παρότι δεν έχει σχεδιαστεί ειδικά για τη δημιουργία του API, η Express περιλαμβάνει επίσης υποστήριξη για template engines όπως οι EJS, Pug και Handlebars, οι οποίες μπορεί να είναι χρήσιμες για τη δημιουργία δυναμικών απαντήσεων HTML σε εφαρμογές που αποδίδονται στην πλευρά του server. Τα template engines επιτρέπουν να ενσωματώνονται δεδομένα σε HTML πρότυπα και να τα αποδίδουν δυναμικά πριν τα σταλθούν στους πελάτες, επιτρέποντάς την δημιουργία για δυναμικές ιστοσελίδες με την Express.

Third-Party Middleware: εκτός από τα ενσωματωμένα middlewares, η Express έχει ένα δυναμικό οικοσύστημα third-party middlewares που επεκτείνουν τη λειτουργικότητά της ακόμη περισσότερο. Υπάρχουν έτοιμοι middlewares για εργασίες όπως authentication, authorization, session management, rate limiting, validation, caching και πολλά άλλα στο npm (Node Package Manager). Αυτά τα third-party middlewares πακέτα τρίτων επιτρέπουν να προθέτονται προηγμένα χαρακτηριστικά στο API της εφαρμογής χωρίς να χρειάζεται να γράφονται από την αρχή.

Αξιοποιώντας αυτά τα εργαλεία και χαρακτηριστικά, μπορούν να δημιουργηθούν αξιόπιστα, κλιμακούμενα και συντηρήσιμα RESTful APIs με την Express. Είτε δημιουργηθεί ένα απλό API για προσωπικό project είτε ένα πολύπλοκο API για μια εφαρμογή μεγάλης κλίμακας, η Express παρέχει όλα όσα χρειάζεται κάποιος προγραμματιστής για να ξεκινήσει και να δημιουργήσει ισχυρά APIs που να ικανοποιούν τις συγκεκριμένες ανάγκες του [31].

3.3.3 Sequelize

Η Sequelize είναι ένα ανοιχτού κώδικα ORM (Object-Relational Mapping) βιβλιοθήκη για την Node.js, η οποία παρέχει έναν απλό τρόπο για την αλληλεπίδραση με βάσεις δεδομένων σχέσεων (SQL databases). Η τεχνολογία ORM επιτρέπει στους προγραμματιστές να δημιουργήσουν, να επεξεργαστούν και να διαχειριστούν δεδομένα χρησιμοποιώντας αντικείμενα στην JavaScript, αντί για να γράφουν απευθείας SQL queries.

Η ιστορία της Sequelize ξεκινάει το 2012, όταν ο Daniel Durante ξεκίνησε το έργο ως μέρος της απαίτησης του για ένα ORM για την JavaScript. Από τότε, η Sequelize έχει εξελιχθεί σε μια από τις κύριες ORM λύσεις για την Node.js, με συνεχείς ενημερώσεις, βελτιώσεις και επεκτάσεις από μια ενεργή κοινότητα προγραμματιστών.

Οι βασικές λειτουργίες της Sequelize περιλαμβάνουν τη δυνατότητα δημιουργίας και διαχείρισης μοντέλων δεδομένων, σχέσεων, καθώς και τη δυνατότητα εκτέλεσης SQL ερωτημάτων μέσω αντικειμένων JavaScript. Επιπλέον, προσφέρει πολλά επιπλέον χαρακτηριστικά όπως ο χειρισμός των συναλλαγών, η υποστήριξη για διάφορους τύπους βάσεων δεδομένων (MySQL, PostgreSQL, SQLite κλπ), η συμβατότητα με την προδιαγραφή ES6 (ECMAScript 2015) και η δυνατότητα διαμόρφωσης με τη χρήση συνήθως χρησιμοποιούμενων προτύπων ανάπτυξης.

Ορισμός Μοντέλων (Model Definition): επιτρέπει στους προγραμματιστές να ορίζουν μοντέλα βάσεων δεδομένων χρησιμοποιώντας κλάσεις JavaScript ή αντικείμενα παρόμοια με JSON. Τα μοντέλα αντιπροσωπεύουν πίνακες στη βάση δεδομένων και καθορίζουν τη δομή των δεδομένων, συμπεριλαμβανομένων των χαρακτηριστικών, των τύπων δεδομένων και των σχέσεων μεταξύ των πινάκων. Με τον ορισμό των μοντέλων, οι προγραμματιστές μπορούν να αλληλεπιδρούν με τη βάση δεδομένων χρησιμοποιώντας γνωστά παραδείγματα αντικειμενοστραφούς προγραμματισμού, κάνοντας πιο εύχρηστες και αποδοτικές τις λειτουργίες των βάσεων δεδομένων.

Επικύρωση Δεδομένων (Data Validation): επιτρέπει στους προγραμματιστές να επιβάλουν περιορισμούς και κανόνες στα δεδομένα που αποθηκεύονται στη βάση δεδομένων. Οι κανόνες επικύρωσης μπορούν να καθοριστούν για το κάθε χαρακτηριστικό ενός μοντέλου, καθορίζοντας απαιτήσεις όπως τύπος δεδομένων, μήκος, μορφή και μοναδικότητα. Η Sequelize επικυρώνει αυτόματα τα δεδομένα σύμφωνα με αυτούς τους κανόνες πριν από την αποθήκευσή τους στη βάση δεδομένων, βοηθώντας στη διατήρηση της ακεραιότητας και της συνέπειας των δεδομένων.

Querying: μια ισχυρή διεπαφή για querying η οποία είναι για την εκτέλεση λειτουργιών CRUD (Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή) σε εγγραφές στην βάση δεδομένων. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν τις query methods της Sequelize για την ανάκτηση, δημιουργία, ενημέρωση και διαγραφή εγγραφών από τη βάση δεδομένων χρησιμοποιώντας απλή σύνταξη JavaScript. Το Sequelize υποστηρίζει μια ευρεία γκάμα επιλογών querying, συμπεριλαμβανομένων των φίλτρων, της ταξινόμησης, της σελιδοποίησης και της συναθροίσεως (aggregation), καθιστώντας εύκολη τη δημιουργία πολύπλοκων και αποδοτικών querying βάσης δεδομένων.

Συναλλαγές (Transactions): μπορεί να ομαδοποιούν πολλαπλές λειτουργίες στις βάσης δεδομένων σε ατομικές μονάδες εργασίας. Οι συναλλαγές εξασφαλίζουν ότι μια σειρά λειτουργιών για την βάση δεδομένων είτε εκτελούνται όλες επιτυχώς είτε αποτυγχάνουν όλες μαζί, βοηθώντας στη διατήρηση της συνέπειας και της ακεραιότητας των δεδομένων. Το API των συναλλαγών της Sequelize παρέχει μεθόδους για την έναρξη, την επιτυχία και την αναίρεση συναλλαγών, προσφέροντας στους προγραμματιστές λεπτομερή έλεγχο.

Συνδέσεις (Associations): επιτρέπει τον ορισμό συνδέσεων μεταξύ πινάκων στις βάσης δεδομένων, όπως οι σχέσεις ένα προς ένα, ένα προς πολλά και πολλαπλά προς πολλαπλά. Οι συνδέσεις επιτρέπουν στους προγραμματιστές να μοντελοποιούν πολύπλοκες σχέσεις δεδομένων στις εφαρμογές τους και να εκτελούν σχετικές ερωτήσεις πιο αποδοτικά.

Hooks: η Sequelize υποστηρίζει συμβάντα lifecycle hooks που επιτρέπουν στους προγραμματιστές να εκτελούν προσαρμοσμένη λογική πριν ή μετά από συγκεκριμένες λειτουργίες της βάσης δεδομένων.

Τα hooks μπορούν να χρησιμοποιηθούν για να εκτελέσουν εργασίες όπως επικύρωση δεδομένων, κανονικοποίηση, καταγραφή και ενεργοποίηση παρενεργειών. Το Sequelize παρέχει μια ποικιλία από hooks για διάφορες λειτουργίες της βάσης δεδομένων.

Migrations: επιτρέπει στους προγραμματιστές να διαχειρίζονται τις αλλαγές στη δομή της βάσης δεδομένων με δομημένο και επαναλαμβανόμενο τρόπο. Τα migrations είναι αρχεία JavaScript που ορίζουν σύνολα αλλαγών στη δομή της βάσης δεδομένων, όπως η δημιουργία ή η τροποποίηση πινάκων, η προσθήκη ή η αφαίρεση στηλών και η εισαγωγή αρχικών δεδομένων. Το API μετακίνησης του Sequelize παρέχει μεθόδους για τη δημιουργία, την εκτέλεση και την αναστροφή μετακινήσεων, κάνοντας εύκολη τη διατήρηση της δομής της βάσης δεδομένων συγχρονισμένη με τη βάση κώδικα της εφαρμογής σε διαφορετικά περιβάλλοντα.

Η Sequelize.js έχει κατακτήσει την εμπιστοσύνη της κοινότητας Node.js και έχει χρησιμοποιηθεί με επιτυχία σε πολλές μεγάλες εφαρμογές και ιστοσελίδες παγκοσμίως. Η συνεχής υποστήριξη και εξέλιξή της καθιστούν μια πολύ καλή επιλογή για την ανάπτυξη Node.js εφαρμογών που απαιτούν αλληλεπίδραση με βάσεις δεδομένων [32].

3.3.4 Zod

Το Zod είναι ένα ισχυρό framework σχήματος δήλωσης και επικύρωσης TypeScript-first schema που στοχεύει στο να απλοποιήσει την επικύρωση δεδομένων και τον ορισμό σχήματος σε έργα TypeScript. Παρέχει μια συνοπτική και εκφραστική σύνταξη για τον καθορισμό σχημάτων δεδομένων και την εκτέλεση επικύρωσης κατά το runtime validation, καθιστώντας πιο εύκολο για τους προγραμματιστές να εξασφαλίσουν τη σωστότητα και την ακεραιότητα των δεδομένων τους.

Η δημιουργία του Zod οδηγήθηκε από την ανάγκη για μια αξιόπιστη και type-safe λύση για την επικύρωση δεδομένων σε έργα TypeScript. Ενώ το TypeScript προσφέρει στατικό έλεγχο τύπου για επικύρωση κατά τη στιγμή της μεταγλώττισης, η επικύρωση κατά τη διάρκεια της εκτέλεσης είναι ακόμη σημαντική για τη διατήρηση της ακεραιότητας των δεδομένων και για τον χειρισμό input του χρήστη ή εξωτερικών πηγών δεδομένων.

Το Zod αναπτύχθηκε από τον Gustav Wengel, έναν μηχανικό λογισμικού με εμπειρία σε ανάπτυξη frontend και backend εφαρμογών. Εμπνευσμένο από υπάρχουσες βιβλιοθήκες επικύρωσης όπως το Joi για την JavaScript και το Pydantic για την Python, ο Wengel αποφάσισε να δημιουργήσει ένα σύγχρονο και αποτελεσματικό πλαίσιο επικύρωσης ειδικά προσαρμοσμένο για την TypeScript.

Το Zod σχεδιάστηκε με το σύστημα των τύπων της TypeScript στο μυαλό, εκμεταλλευόμενο την αυτόματη ενέργεια εντοπισμού τύπων και generics για να παρέχει μια ομαλή και εύκολη εμπειρία επικύρωσης. Επιτρέπει στους προγραμματιστές να ορίσουν σύνθετα σχήματα δεδομένων χρησιμοποιώντας τη σύνταξη τύπων της TypeScript, συμπεριλαμβανομένων πρωτογενών τύπων, μορφών αντικειμένου, πίνακων, ένωσης και άλλων.

Ένας από τους βασικούς σχεδιαστικούς κανόνες του Zod είναι η έμφασή του στην εμπειρία του προγραμματιστή και την εργονομία. Η βιβλιοθήκη προσπαθεί να παρέχει μια απλή και εύκολη στη χρήση διεπαφή για τον καθορισμό σχημάτων και την εκτέλεση επικύρωσης, ελαχιστοποιώντας τον περιττό κώδικα και την αντιμετώπιση κενών στην κατανόηση τους.

Η αρχιτεκτονική του Zod βασίζεται σε έναν πυρήνα συνόλου τύπων και συναρτήσεων TypeScript που επιτρέπουν τον ορισμό σχημάτων, την επικύρωση, την ανάγκη και τη μετατροπή. Παρέχει μια ομαλή και συνεκτική διεπαφή για τον ορισμό σχημάτων, τη σύνδεση μεθόδων επικύρωσης και τη διαχείριση

σφαλμάτων, επιτρέποντας στους προγραμματιστές να εκφράσουν σύνθετη λογική επικύρωσης με κατανοητό και σύντομο τρόπο [33].

Παρακάτω είναι μερικά από τα βασικά χαρακτηριστικά και εργαλεία που προσφέρει το Zod για τη δημιουργία αντικειμένων με ασφαλή τύπους:

- Ορισμός Σχήματος (Schema Definition): Το Zod μπορεί να ορίσει σχήματα για τις δομές δεδομένων σας χρησιμοποιώντας ένα fluent API. Μπορεί να σχηματίσει κάνεις απλούς τύπους όπως συμβολοσειρές, αριθμούς, λογικές τιμές, καθώς και πιο περίπλοκα εμπειρία αντικειμένων, πίνακες, enums, ενώσεις και άλλα.
- Type Inference: Μόλις ορίσετε ένα σχήμα χρησιμοποιώντας το Zod, αυτό αυτόματα επιστρέφει τους τύπους TypeScript που αντιστοιχούν στο σχήμα σας. Αυτό σημαίνει ότι λαμβάνετε ισχυρό στατικό έλεγχο τύπων για τα δεδομένα σας χωρίς να γράφετε περιττές αναφορές τύπων.
- Επικύρωση Δεδομένων (Data Validation): Το Zod παρέχει μεθόδους για την επικύρωση των εισερχόμενων δεδομένων έναντι των ορισμένων σχημάτων. Εξασφαλίζουν ότι τα δεδομένα που λαμβάνει ή χειρίζεται κάποιος στην εφαρμογή, να πληρούν την αναμενόμενη δομή και τους περιορισμούς.
- Type Casting and Parsing: αναλύει τα input δεδομένα σε αντικείμενα strongly typed objects με βάση τα ορισμένα σχήματα σας. Υποστηρίζει επίσης casting type για να αναγκάσει τα δεδομένα σε αναμενόμενους τύπους, όπως να χειρίζεται σενάρια όπως η ανάλυση συμβολοσειρών σε αριθμούς ή ημερομηνίες.
- Χειρισμός Σφαλμάτων (Error Handling): Όταν η επικύρωση αποτυγχάνει, το Zod παρέχει λεπτομερείς μηνύματα σφάλματος που υποδεικνύουν τι πήγε στραβά και πού. Αυτό βοηθά στο debugging και στην παροχή σημαντικών πληροφοριών στους χρήστες σχετικά με τη μη έγκυρη input.
- Ασύγχρονη Επικύρωση (Async Validation): Το Zod υποστηρίζει ασύγχρονη επικύρωση, επιτρέποντάς την εκτέλεση επικύρωσης που περιλαμβάνουν ασύγχρονες λειτουργίες όπως η λήψη δεδομένων από εξωτερικό API ή βάση δεδομένων.
- Προσαρμοσμένοι Επικυρωτές (Custom Validators): Μπορείτε να ορίσετε προσαρμοσμένους επικυρωτές χρησιμοποιώντας το API του Zod, επιτρέποντάς να επιβάλλει κανόνες επικύρωσης συγκεκριμένοι στην εφαρμογή πέρα από απλούς περιορισμούς σχήματος.
- Σύνθεση Σχήματος (Schema Composition): συνδιάζει πολλαπλά σχήματα σε μεγαλύτερα σχήματα. Αυτό επιτρέπει την επαναχρησιμοποίηση κώδικα και το modularization της λογικής επικύρωσης.
- Μετασχηματισμός Σχήματος (Schema Transformation): Το Zod παρέχει μεθόδους για τον μετασχηματισμό και τη διαχείριση των σχημάτων, όπως mapping πάνω από τιμές σχήματος ή η εξαγωγή συγκεκριμένων πεδίων από αντικείμενα.
- Runtime Validation: Το Zod μπορεί να χρησιμοποιηθεί για runtime validation σε περιβάλλοντα Node.js ή περιβάλλοντα περιήγησης, εξασφαλίζοντας ότι η εφαρμογή παραμένει ανθεκτική απέναντι σε απροσδόκητα δεδομένα.

3.3.5 Swagger

Το Swagger είναι ένα λογισμικό ανοιχτού κώδικα που υποστηρίζεται από το OpenAPI και επιτρέπει στους προγραμματιστές να περιγράψουν, να τεκμηριώσουν και να καταναλώσουν RESTful APIs. Παρέχει ένα σύνολο εργαλείων για το σχεδιασμό, την ανάπτυξη και την τεκμηρίωση APIs με έναν

τυποποιημένο τρόπο, καθιστώντας ευκολότερο για τους προγραμματιστές την κατανόηση και την αλληλεπίδραση με APIs σε διάφορες πλατφόρμες και γλώσσες προγραμματισμού [34].

Η ιστορία του Swagger ξεκινά με τον Tony Tam, έναν μηχανικό λογισμικού στην Wordnik, που αναγνώρισε την ανάγκη για έναν τυποποιημένο τρόπο για την τεκμηρίωση και την αλληλεπίδραση με RESTful APIs. Ξεκίνησε την ανάπτυξη του Swagger το 2010 ως ένα εσωτερικό έργο στην Wordnik για να αντιμετωπίσει αυτές τις προκλήσεις. Το Swagger αρχικά ξεκίνησε ως ένα απλό JSON-based format για την περιγραφή των RESTful APIs, παρέχοντας μεταδεδομένα σχετικά με τα endpoints, τις παραμέτρους αιτήσεων, τις μορφές απόκρισης και τις μεθόδους ταυτοποίησης.

Καθώς το Swagger έγινε δημοφιλές μέσα στην κοινότητα των προγραμματιστών, κυκλοφόρησε ως ένα ανοιχτού κώδικα έργο το 2011 με άδεια του Apache 2.0. Αυτή η κίνηση επέτρεψε σε προγραμματιστές από όλο τον κόσμο να συνεισφέρουν στην ανάπτυξή του, με αποτέλεσμα τη δημιουργία ενός δυναμικού οικοσυστήματος γύρω από το Swagger.

Το 2015, η προδιαγραφή του Swagger προσφέρθηκε δωρεάν στην OpenAPI, ένα μέρος με επιχειρηματίες και open-source contributors αφιερωμένοι στη δημιουργία, εξέλιξη και προώθηση του Προτύπου OpenAPI (OAS). Ο στόχος του OpenAPI ήταν η τυποποίηση της μορφής που χρησιμοποιείται για την περιγραφή των RESTful APIs, κάνοντας ευκολότερη τη συνεργασία και την ενσωμάτωση των APIs σε διαφορετικές πλατφόρμες και εργαλεία.

Η προδιαγραφή του Swagger χρησίμευσε ως βάση για το OpenAPI, το οποίο είχε στόχο να παρέχει ένα γλωσσο-ανεξάρτητο μορφότυπο περιγραφής για τα RESTful APIs. Η OpenAPI χτίζει πάνω στις ιδέες που παρουσιάζει το Swagger, προσθέτοντας νέα χαρακτηριστικά, διευκρινίσεις και βελτιώσεις με βάση τα σχόλια της κοινότητας και τις βέλτιστες πρακτικές της βιομηχανίας.

Το Swagger προσφέρει αρκετά βασικά χαρακτηριστικά και οφέλη για προγραμματιστές και οργανισμούς που εμπλέκονται στην ανάπτυξη APIs:

Τεκμηρίωση του API: δημιουργία αυτόματης τεκμηρίωσης του API από τις προδιαγραφές του API. Αυτή η τεκμηρίωση περιλαμβάνει λεπτομέρειες σχετικά με τα endpoints, τις παραμέτρους αιτήσεων, τις μορφές απόκρισης, τους κωδικούς σφαλμάτων και τις απαιτήσεις ταυτοποίησης.

Εργαλεία Σχεδιασμού του API: Το Swagger παρέχει ένα σύνολο εργαλείων για το σχεδιασμό των APIs, συμπεριλαμβανομένων οπτικών επεξεργαστών, code generators και ελεγκτών.

Client SDK Generation: είναι για διάφορες γλώσσες προγραμματισμού και φτιάχνετε σύμφωνα από προδιαγραφές του API. Αυτά τα Client SDK παρέχουν έτοιμες βιβλιοθήκες που αφαιρούν τις πολυπλοκότητες της κατάθεσης αιτημάτων HTTP και της χειρισμού των αποκρίσεων του API.

Δοκιμή και Επικύρωση του API: Το Swagger διευκολύνει τη δοκιμή και την επικύρωση του API παρέχοντας εργαλεία για την αποστολή αιτημάτων, την επικύρωση απαντήσεων και τον έλεγχο της συμμόρφωσης με τις προδιαγραφές του API.

3.4 Typescript

Η TypeScript είναι μια προγραμματιστική γλώσσα ανοιχτού κώδικα που αναπτύχθηκε από τη Microsoft. Είναι ένα υπερσύνολο της JavaScript, που σημαίνει ότι κάθε έγκυρος κώδικας JavaScript είναι επίσης έγκυρος κώδικας TypeScript. Η TypeScript προσθέτει προαιρετικά static types στην JavaScript, επιτρέποντας στους προγραμματιστές να ορίζουν τύπους για μεταβλητές, παραμέτρους συναρτήσεων, επιστρεφόμενες τιμές και περισσότερα. Αυτές οι αναφορές τύπου χρησιμοποιούνται από τον μεταγλωττιστή της TypeScript για την ανίχνευση σφαλμάτων σχετικά με τον τύπο κατά τη

διάρκεια της ανάπτυξης, προσφέροντας βελτιωμένη ποιότητα κώδικα και καλύτερη υποστήριξη εργαλείων σε σύγκριση με την παραδοσιακή JavaScript [35].

Στατική Τυποποίηση (Static Typing): καθορίζει τους τύπους των μεταβλητών και άλλων κατασκευών στον κώδικά τους. Αυτό βοηθά στην ανίχνευση σφαλμάτων που σχετίζονται με τον τύπο στα πρώιμα στάδια της διαδικασίας ανάπτυξης, οδηγώντας σε πιο ανθεκτικό και εύκολο στη συντήρηση κώδικα.

Βελτιωμένα Εργαλεία: η TypeScript ενσωματώνεται άψογα με δημοφιλείς εφαρμογές επεξεργασίας κώδικα, όπως το Visual Studio Code, παρέχοντας χαρακτηριστικά όπως συμπλήρωση κώδικα, ευρετήριο τύπων και αυτόματη αναδιάρθρωση. Αυτό βελτιώνει την παραγωγικότητα του προγραμματιστή και καθιστά ευκολότερη την πλοήγηση και την κατανόηση πολύπλοκων βιβλιοθηκών κώδικα.

Βελτιωμένη Αναγνωσιμότητα: οι αναφορές τύπων λειτουργούν ως τεκμηρίωση για τον κώδικα, κάνοντας ευκολότερη την κατανόηση της προοριζόμενης χρήσης των συναρτήσεων και των δομών δεδομένων. Αυτό βελτιώνει την αναγνωσιμότητα του κώδικα και μειώνει την πιθανότητα σφαλμάτων που προκαλούνται από παρανοήσεις ή εσφαλμένες υποθέσεις.

Συντηρησιμότητα του Κώδικα: προσθέτοντας αναφορές τύπου σε κώδικα JavaScript, η TypeScript βοηθά στην επιβολή σαφών διεπαφών και συμβάσεων μεταξύ διαφορετικών τμημάτων της βάσης κώδικα. Αυτό καθιστά πιο εύκολη την αναδιάρθρωση και τη συντήρηση του κώδικα με τον χρόνο, καθώς οι αλλαγές μπορούν να γίνουν με την πεποίθηση ότι δεν θα καταστρέψουν ακούσια άλλα τμήματα της εφαρμογής.

Συμβατότητα Οικοσυστήματος: η TypeScript σχεδιάστηκε να είναι συμβατή με υπάρχουσες βιβλιοθήκες και frameworks της JavaScript. Αυτό σημαίνει ότι οι προγραμματιστές μπορούν να αξιοποιήσουν τον πλούτο του οικοσυστήματος των πακέτων JavaScript, ενώ εξακολουθούν να απολαμβάνουν τα οφέλη της στατικής τυποποίησης και άλλων χαρακτηριστικών της TypeScript.

Όταν αναπτύσσονται εφαρμογές με την React και με την TypeScript, οι προγραμματιστές μπορούν να ορίζουν διεπαφές ή τύπους για props και state, διασφαλίζοντας ότι τα συστατικά λαμβάνουν τους σωστούς τύπους δεδομένων και αποτρέποντας συνηθισμένα σφάλματα όπως η μετάδοση μη έγκυρων props ή η πρόσβαση σε μεταβλητές που δεν έχουν καθοριστεί. Επιπλέον, οι δυνατότητες ανάκλασης τύπου της TypeScript βοηθούν στην απλοποίηση της διαδικασίας ανάπτυξης με την αυτόματη εύρεση τύπων με βάση τη χρήση, μειώνοντας την ανάγκη για σαφείς αναφορές τύπου σε πολλές περιπτώσεις. Η TypeScript βελτιώνει επίσης την εμπειρία του προγραμματιστή και με την JSX, τη συντακτική επέκταση την React για τον καθορισμό των components. Ο μεταγλωττιστής της TypeScript κατανοεί τη σύνταξη JSX και μπορεί να ελέγξει τους τύπους των JSX εκφράσεων, βοηθώντας στην ανίχνευση σφαλμάτων όπως η αντιστοίχιση αταίριαστων τύπων γνωρισμάτων ή η απουσία απαιτούμενων props.

Η TypeScript μπορεί να ενσωματωθεί απροβλημάτιστα σε Node.js project, παρέχοντας τα ίδια οφέλη της στατικής τυποποίησης και της βελτιωμένης υποστήριξης εργαλείων που προσφέρει για την ανάπτυξη στο frontend. Όταν χρησιμοποιείται η TypeScript με την Node.js, οι προγραμματιστές μπορούν να ορίζουν τύπους για παραμέτρους συναρτήσεων, τιμές επιστροφής και άλλες κατασκευές, βοηθώντας στην ανίχνευση σφαλμάτων στα πρώιμα στάδια της διαδικασίας ανάπτυξης και βελτιώνοντας την ποιότητα του κώδικα. Παρέχει επίσης υποστήριξη για σύγχρονες λειτουργίες της JavaScript όπως το async/await, τα modules και η ES6 σύνταξη, επιτρέποντας στους προγραμματιστές να γράφουν καθαρότερο και πιο εκφραστικό κώδικα. Επιπλέον, όπως και με την React, υπάρχει και εδώ η συμβατότητα της TypeScript με υπάρχουσες βιβλιοθήκες και frameworks της Node.js.

Κεφάλαιο 4ο: Σχεδίαση και Υλοποίηση του HellenicCSResearch

4.1 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις αποτελούν ένα θεμελιώδες κεφάλαιο στον σχεδιασμό και την ανάπτυξη κάθε εφαρμογής λογισμικού. Αυτές οι απαιτήσεις καθορίζουν τις λειτουργίες και τις δυνατότητες που πρέπει να διαθέτει η εφαρμογή για να είναι λειτουργική και χρήσιμη για τους τελικούς χρήστες της. Οι λειτουργικές απαιτήσεις περιγράφουν το τι μπορεί να κάνει η εφαρμογή, πώς μπορεί να αλληλεπιδρά με τους χρήστες και ποιες λειτουργίες πρέπει να εκτελεί σε διάφορες συνθήκες. Αυτό μπορεί να περιλαμβάνει τη δυνατότητα διαχείρισης δεδομένων, τη διαχείριση των χρηστών, την απόδοση εργασιών, την αναφορά δεδομένων, και πολλά άλλα. Για να καθοριστούν αυτές οι απαιτήσεις με ακρίβεια, συχνά χρησιμοποιούνται διάφορες τεχνικές, όπως οι user stories, οι οποίες περιγράφουν τις λειτουργίες από την οπτική γωνία του χρήστη, ή άλλα μοντέλα απαιτήσεων λογισμικού. Τέλος, οι λειτουργικές απαιτήσεις δεν είναι στατικές· μπορεί να υποστούν αλλαγές κατά τη διάρκεια του κύκλου ζωής του έργου. Είναι σημαντικό να διατηρούνται ενημερωμένες και να προσαρμόζονται στις νέες ανάγκες και απαιτήσεις του χρήστη, καθώς και σε ενδεχόμενες αλλαγές στο περιβάλλον λειτουργίας της εφαρμογής. Παρακάτω υπάρχουν οι λειτουργικές απαιτήσεις του HellenicCSResearch.

Κοινά Στοιχεία (Προσαρμογή και Χρήστη):

- Αλλαγή Θέματος: Ο χρήστης έχει τη δυνατότητα να επιλέξει μεταξύ dark και light θέματος για την εφαρμογή
- Προσαρμογή Πίνακα: Ο χρήστης μπορεί να προσαρμόσει το ύψος του πίνακα για να ταιριάζει καλύτερα στις προτιμήσεις του.
- Εκκαθάριση Φίλτρων: Υπάρχει ένα κουμπί εκκαθάρισης που επιτρέπει στον χρήστη να επαναφέρει όλα τα φίλτρα στην αρχική τους κατάσταση.
- Εμφάνιση Chips για Επιλεγμένα Φίλτρα: Τα επιλεγμένα φίλτρα παρουσιάζονται σε μορφή chips για εύκολη πρόσβαση και διαχείριση από τον χρήστη.

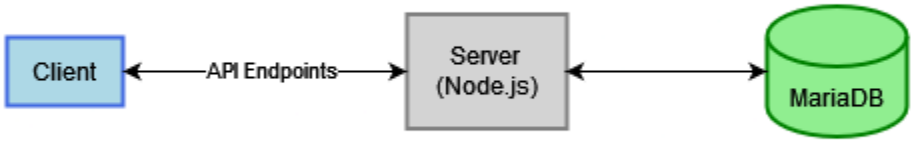
Σελίδα Citations:

- Εμφάνιση Μηνύματος Επιλογής Φίλτρων: Όταν ο χρήστης επισκέπτεται τη σελίδα Citations για πρώτη φορά, εμφανίζεται ένα μήνυμα που τον καλεί να επιλέξει κάποιο τμήμα για την προβολή των στατιστικών. Προβολή των διαθέσιμων φίλτρων: εύρος ετών των δημοσιεύσεων, δημοσιεύσεις χωρίς χρονολογία, τμήματα, ακαδημαϊκή θέση.
- Στατιστικά Σχετικά με τα Επιλεγμένα Departments: Προβολή γενικών στατιστικών για τα επιλεγμένα τμήματα (όπως συνολικός αριθμός αναφορών). Προβολή γραφήματος πίτας που δείχνει το ποσοστό συνολικού αριθμού ανά ακαδημαϊκή θέση.
- Στατιστικά για το Προσωπικό των Επιλεγμένων Departments: Προβολή στατιστικών σχετικά με το προσωπικό των επιλεγμένων τμημάτων (π.χ. h-index).
- Προβολή Πίνακα Δημοσιεύσεων/Αναφορών: Προβολή πίνακα με τις αναφορές/δημοσιεύσεις, με δυνατότητα ταξινόμησης και φιλτραρίσματος βάσει των επιλεγμένων κριτηρίων.
- Διάγραμμα Αναφορών/Δημοσιεύσεων ανά Χρονιά: Εμφάνιση διαγράμματος που δείχνει τον αριθμό των αναφορών ή δημοσιεύσεων ανά χρονιά ή και των δύο. Εναλλαγή γίνεται μέσω toggle button.

Σελίδα Departments:

- Εμφάνιση Μηνύματος Επιλογής Ακαδημαϊκής Θέσης: Όταν ο χρήστης επισκέπτεται τη σελίδα Departments για πρώτη φορά, εμφανίζεται ένα μήνυμα που τον καλεί να επιλέξει κάποια ακαδημαϊκή θέση προκειμένου να εμφανιστούν στατιστικά στοιχεία. Προβολή των διαθέσιμων φίλτρων: εύρος ετών των δημοσιεύσεων, δημοσιεύσεις χωρίς χρονολογία, ακαδημαϊκή θέση.
- Προβολή Στατιστικών ανά Τμήμα: Μετά την επιλογή της ακαδημαϊκής θέσης από τον χρήστη, εμφανίζεται ένας πίνακας με στατιστικά στοιχεία για κάθε τμήμα. Τα στατιστικά στοιχεία είναι δυναμικά και ανανεώνονται ανάλογα με τις επιλογές του χρήστη.
- Εμφάνιση Διαγραμμάτων με Στατιστικά Στοιχεία: Εάν ο χρήστης επιλέξει μια γραμμή από τον πίνακα, τότε εμφανίζονται τρία διαγράμματα με στατιστικά στοιχεία. Το πρώτο διάγραμμα παρουσιάζει πίτες με τα συνολικά citations ανά ακαδημαϊκή θέση του τμήματος. Το δεύτερο διάγραμμα παρουσιάζει πίτες με τα συνολικά publications ανά ακαδημαϊκή θέση του τμήματος. Το τρίτο διάγραμμα δείχνει τα citations και τα publications του προσωπικού του τμήματος. Τα διαγράμματα εμφανίζουν τα στατιστικά στοιχεία που έχει επιλέξει ο χρήστης από τα φίλτρα.

4.2 Αρχιτεκτονική της εφαρμογής



Εικόνα 4.1: Διάγραμμα αρχιτεκτονικής της εφαρμογής

Η εφαρμογή μας ακολουθεί μια αρχιτεκτονική που διασφαλίζει αποτελεσματική επικοινωνία μεταξύ του client, του backend server και της βάσης δεδομένων MariaDB. Η παραπάνω εικόνα δείχνει τον κύκλο ζωής και την αρχιτεκτονική της εφαρμογής. Ο client ανταλλάσσει δεδομένα με τον backend server μέσω API endpoints. Χρησιμοποιεί τις παρεχόμενες υπηρεσίες για να αποκτήσει πρόσβαση στα δεδομένα της εφαρμογής μας. Ο server λειτουργεί ως μεσολαβητής μεταξύ του client και της βάσης δεδομένων. Αφού λάβει αιτήματα από τον client μέσω των endpoints, αποκτά πρόσβαση στην βάση δεδομένων για να ανακτήσει τα δεδομένα που απαιτούνται. Η βάση δεδομένων MariaDB αποτελεί τον κύριο αποθηκευτικό χώρο των δεδομένων.

4.3 Υλοποίηση του backend

Για την δημιουργία του backend χρησιμοποιήθηκαν οι Node.js, Express, Sequelize και Typescript που αναλύθηκαν στο δεύτερο κεφάλαιο. Για να λειτουργήσει ο server, η Node,js εκτελεί το index.ts αρχείο το οποίο φορτώνει τις μεταβλητές περιβάλλοντος από τα αρχεία ".env" ΄χάρης την βιβλιοθήκη dotenv, και δημιουργεί ένα νέο αντικείμενο της κλάσης Server, που θα αναλυθεί παρακάτω. Στην

Εικόνα 2.3 *Server class – Properties & Constructor*, βλέπουμε ότι έχει τα εξής τρία properties που είναι προσβάσιμα μόνο μέσα στην ίδια κλάση:

- app: δημιουργεί μια νέα Express εφαρμογή.
- port: χρησιμοποιείται για την αποθήκευση ενός αριθμού που αντιπροσωπεύει μια πύλη (port).
- host: χρησιμοποιείται για να αποθηκεύσετε το όνομα του server.

Στον constructor οι μεταβλητές port και host παίρνουν τις αντίστοιχες τιμές από το .env file εάν υπάρχουν, αλλιώς ορίζονται κάποιες προκαθορισμένες τιμές.

```
class Server {
    10 references
    private app = express();
    3 references
    private port: number;
    3 references
    private host: string;

    1 reference
    constructor() {
        this.port = (process.env.PORT || 8888) as number;
        this.host = process.env.HOST || 'localhost'
        this.listen();
        this.routes();
        this.dbConnect();
        this.middlewares();
        runAssociations();
    }
```

Εικόνα 4.2: Server class – Properties & Constructor

Η μέθοδος listen() καλεί την μέθοδο listen της Express ώστε να ξεκινήσει τον server και να ‘ακούει’ πότε θα γίνει η σύνδεση στην διεύθυνση και πύλη που περνάνε σαν παραμέτρους στην μέθοδο. Μόλις γίνει η σύνδεση τότε καλή την μέθοδο swaggerDocs που περνάει σαν παράμετρο την Express.

```
// Listens for a connection
1 reference
listen(): void {
    this.app.listen(this.port, this.host, (): void => {
        console.log(`Application runs at host: ${this.host} and port: ${this.port}`);
        swaggerDocs(this.app);
    });
}
```

Εικόνα 4.3: Server class – listen method

```
const options: swaggerJSDoc.Options = {
    definition: {
        openapi: "3.1.0",
        info: {
            title: "OMEA API Docs",
            version: pjson.version
        },

    },
    apis: ["./src/routes/*.route.ts", "./src/api/common.ts"]
};

const swaggerSpec = swaggerJSDoc(options);

const swaggerDocs = (app: Express) => {
    console.log(`Docs available at http://${process.env.HOST}:${process.env.PORT}/api/docs`);
    app.use('/api/docs', swaggerUI.serve, swaggerUI.setup(swaggerSpec));

    app.get('docs.json', (req: Request, res: Response) => {
        res.setHeader("Content-Type", "application/json");
        res.send(swaggerSpec);
    })

};

2 references
export default swaggerDocs;
```

Εικόνα 4.4: Init Swagger Docs API

Παραπάνω στην εικόνα, φαίνεται το πως γίνεται αρχικοποίηση του swaggerJSDoc με τα options που ορίζονται πάνω από αυτό. Μία σημαντική λεπτομέρεια είναι η λίστα apis, που ορίζει κάποιες διαδρομές αρχείων όπου θα διαβάζει για τα apis. Στην συνέχεια φτιάχνει έναν middleware ο οποίος ακούει στην διαδρομή '/api/docs', που εξυπηρετεί τα στατικά αρχεία του Swagger UI περνώντας τις προδιαγραφές που ορίστηκαν πριν. Έπειτα, φτιάχνει ένα route για GET αιτήσεις στην διεύθυνση '/docs.json' και γυρνάει τις προδιαγραφές για το Swagger.

Το επόμενο που καλεί ο constructor είναι το routes. Η μέθοδος διαχειρίζεται όλες τις διαδρομές τις εφαρμογής. Τα δύο σημαντικά πράγματα που εκτελεί είναι να μπορεί ο server να μπορεί να διαβάζει JSON δεδομένα που έρχονται στο request body, και δεύτερον να ενεργοποιεί το CORS (Cross-Origin Resource Sharing) για όλες τις διαδρομές. Είναι ένα χαρακτηριστικό ασφαλείας που υλοποιείται από τα προγράμματα περιήγησης ιστού για τον έλεγχο των request που υποβάλλονται από διαφορετικές πηγές προέλευσης με δυνατότητες να οριστούν κανόνες σχετικά με το ποια origins, μέθοδοι και headers επιτρέπονται για την πρόσβαση σε πόρους στο διακομιστή. Τέλος φτιάχνει τους τέσσερις

βασικούς middlewares, οι οποίοι θα επεξεργάζονται όλα τα endpoints, και ένα route στο '/' που απλά εμφανίζει ένα μήνυμα για να δείξει ότι λειτουργεί ο server. Πιο παρακάτω περιγράφεται ένας από τους τέσσερις κύριους middleware που χρησιμοποιεί η εφαρμογή.

```
// Handles all endpoints
1 reference
routes(): void {
    // Parse the body
    this.app.use(express.json());

    // Cors must be before any route
    // Cors
    this.app.use(cors());

    this.app.get('/', (req: Request, res: Response): void => {
        res.json({
            msg: "It's all good man!"
        });
    });
    this.app.use('/api/departments', routesDepartment);
    this.app.use('/api/publications', routesPublications);
    this.app.use('/api/academicStaff', routesAcademicStaff);
    this.app.use('/api/yearsRange', routesyearsRange);
}
```

Εικόνα 4.5: Server class – routes method

Η τρίτη μέθοδος είναι η dbConnect που επιμελεί την σύνδεση με την βάση δεμένων MariaDB. Καλεί ασύγχρονα την db.authenticate() που βρίσκεται σε μία try/catch, η οποία τεστάρει μία σύνδεση με την βάση προσπαθώντας να ελέγξει την ταυτότητα. Και στις δύο περιπτώσεις εμφανίζει ανάλογο μήνυμα. Το db είναι μία Sequelize class η οποία απεικονίζετε στην Εικόνα 4.7 *Connect to DB via Sequelize*.

```
// Checks if the connection at database was successful or not
1 reference
async dbConnect(): Promise<void> {
    try {
        await db.authenticate();
        console.log('Connected to the database!');
    } catch {
        // Sequelize will handle the reconnect by itself.
        console.log('Unable to connect to the database.');
    }
}
```

Εικόνα 4.6: Server class – dbConnect method

Η db φτιάχνει μία νέα κλάση Sequelize που παίρνει σαν παραμέτρους το όνομα της βάσης, όνομα χρήστη, κωδικό χρήστη, ενώ η τέταρτη είναι ένα object που μπορεί να βάλεις πολλούς παραμέτρους αλλά οι κύριες είναι τι τύπου βάση είναι και φυσικά τον όνομα του server και τον αριθμό πύλης που ακούει.

```
import { Sequelize } from "sequelize";

const sequelize = new Sequelize(process.env.DB_NAME || "citations", process.env.DB_USER || '', process.env.DB_PASSWORD || '', {
    host: process.env.DB_HOST || '',
    port: (process.env.DB_PORT || 3306) as number,
    dialect: 'mariadb',
    define: {
        // Stop the auto-pluralization performed by Sequelize. This way will infer the table name to be equal to the model name
        freezeTableName: true,
        // Does not let Sequelize to add automatically the fields createdAt and updatedAt
        timestamps: false
    },
    // For log the sql queries
    logging: false,
});

16 references
export default sequelize;
```

Εικόνα 4.7: Connect to DB via Sequelize

Η προ-τελευταία μέθοδος είναι για τους middlewares. Πρέπει να καλούνται τελευταίοι από τα routes που ορίζομαι, επειδή όμως όπως βλέπουμε στην Εικόνα 4.7 *Server class – routes method*, έχει γίνει η χρήση middlewares για να “τυλίξουν” τα routes σε κάτι πιο γενικό, πρέπει πάλι να καλεστεί μετά από αυτά. Ο συγκεκριμένος χειρίζεται τα σφάλματα που μπορεί να προκύψουν π.χ. να μην γίνει η σύνδεση με την βάση ή ο χρήστης να έβαλε λάθους παραμέτρους σε μία GET διεύθυνση. Θα γίνει η ανάλυση του errorHandler στο παράδειγμα το τι γίνεται όταν καλεστεί ένα endpoint.

Εικόνα 4.8: Server class – middlewares method

Τελευταία μέθοδος είναι η runAssociations. Έχει να κάνει με την Sequelize και ορίζει τις συσχετίσεις μεταξύ των μοντέλων, όπως είναι και στην βάση. Με αυτές τις συσχετίσεις, ορίζονται οι σχέσεις μεταξύ των διαφόρων πινάκων στη βάση δεδομένων, όπως η σχέση 1-1, 1-N, και N-1. Αυτές οι σχέσεις μπορούν να χρησιμοποιηθούν για την ανάκτηση δεδομένων από διαφορετικούς πίνακες και τη διευκόλυνση της διαδικασίας ανάγνωσης και εγγραφής δεδομένων στη βάση.

Εικόνα 4.9: Database Model Associations in Sequelize

Στην Εικόνα 4.9 *Creating a Publication Model in Sequelize,* δείχνει στο πως ορίζεται ένα μοντέλο. Η πρώτη παράμετρος της define είναι το όνομα του πίνακα που υπάρχει στην βάση, ενώ η δεύτερη περιέχει τα πεδία και τις σχετικές πληροφορίες:

- id: αντιστοιχεί στο πεδίο “gsid” στη βάση δεδομένων, είναι τύπου string με μέγιστο μήκος 255 χαρακτήρες. Είναι πρωτεύων κλειδί και αναφέρεται στο πεδίο “gsid” του πίνακα Dep. Το CASCADE δηλώνει ότι εάν αλλάξει είτε διαγραφεί το κλειδί που κάνει αναφορά στον πίνακα Dep, τότε θα γίνει το αντίστοιχο και για το Publication
- year: αντιστοιχεί στο πεδίο “cyear”, είναι τύπου integer. Επίσης είναι και αυτό πρωτεύων κλειδί
- counter: αντιστοιχεί στο πεδίο “cyear”, είναι τύπου integer και δεν επιτρέπει τιμές null.

```
const Publications = db.define<IPublications>('publications', {
    id: {
        field: 'gsid',
        type: DataTypes.STRING(255),
        allowNull: false,
        primaryKey: true,
        references: {
            model: Dep,
            key: 'gsid'
        },
        onDelete: 'CASCADE',
        onUpdate: 'CASCADE'
    },
    year: {
        field: 'cyear',
        type: DataTypes.INTEGER,
        allowNull: false,
        primaryKey: true
    },
    counter: {
        field: 'counter',
        type: DataTypes.INTEGER,
        allowNull: false
    }
});

13 references
export default Publications;
```

Εικόνα 4.10: Creating a Publication Model in Sequelize

Σε αυτό το σημείο ο server τρέχει, είναι συνδεδεμένος με την βάση και περιμένει μέσω των endpoints να εξυπηρετήσει. Στην συνέχεια θα δούμε πως λειτουργεί ένα endpoint, πως γίνεται το caching, τα validators και η διαχείριση των σφαλμάτων. Στην Εικόνα 4.11 *Declaration for Academic Staff Positions* βλέπουμε τον middleware με διαδρομή " /api/academicStaff" από την Εικόνα 4.5 *Server class – routes method*. Δημιουργεί δύο endpoints ως GET request. Η διαδρομή για το endpoint δεν είναι μόνο το "/positions" αλλά μαζί με το path του middleware, δηλαδή θα είναι "/api/academicStaff /positions". Όλοι οι παράμετροι μετά από τον πρώτο είναι middlewares, ο τελευταίος είναι ο controller του endpoint ενώ όλοι οι άλλοι, εάν υπάρχουν, στην εφαρμογή είναι middlewares για να ελέγχουν κάποια δεδομένα για τον controller άμα υπάρχουν στην cache.

```
const router = Router();

/**...
router.get('/positions', getCacheAllPositions, getPositions);

/** ...
router.get('/positionsSumByDepartment', getCacheAllPositions, getCacheDepartmentsID, getPositionsCountByDepartment);

2 references
export default router;
```

Εικόνα 4.11: Routers Declaration for Academic Staff Positions

Οι middlewares εκτελούνται με την σειρά, δηλαδή πρώτα οι middlewares για τα cache δεδομένα. Στην Εικόνα 4.12 *Routers Declaration for Academic Staff Positions*, υπάρχει μία Until μέθοδος tryCatch, που εάν γυρίσει error, αντί να πάει αυτόματα στον επόμενο middleware, θα καλέσει την errorHandle που είδαμε στην Server class. Το middleware ελέγχει εάν υπάρχει στην cache τα δεδομένα. Στην περίπτωση που υπάρχουν τότε απλά στο req.cache προσθέτει τα cached δεδομένα και καλή το επόμενο middleware. Τώρα εάν δεν υπάρχουν καλεί την getAllPositions που κάνει ένα query στην βάση για να γυρίσει τις ακαδημαϊκές θέσεις, έπειτα βάζε τα δεδομένα στην cache μέσω του put και οι παράμετροι είναι: το key, τα δεδομένα και πόση ώρα να μείνουν στην cache.

```
export const getCacheAllPositions = tryCatch(async (req: omeaCitationsReqBody<unknown>,
    res: omeaCitationsRes<string[]>, next: NextFunction) => {
    const cachedData: string[] = cache.get(cacheKeysEnum.Position);
    if (!cachedData) {
        const result = await getAllPositions();
        cache.put(cacheKeysEnum.Position, result, cacheTime);
        reqCache.position = result;
    }
    req.cache = reqCache;
    next();
}, true);
```

Εικόνα 4.12: Routers Declaration for Academic Staff Positions

Τέλος θα δούμε ένα απλό controller. Δεν περιμένει κάποιο request οπότε κάνε κατευθείαν το query στην βάση. Χρησιμοποιεί το μοντέλο Publications και γυρνάει όλα τα δεδομένα. Εφόσον τελειώσει το query και δεν έχει κάποιο σφάλμα, θα αποστείλει μια json απάντηση πίσω με τα δεδομένα που πήρε από το query, με code status 200 και ένα μήνυμα ότι δεν είχε κάποιο σφάλμα.

```
export const getPublications = tryCatch(async (req: omeaCitationsReqBody<unknown>,
    res: omeaCitationsRes<IPublications[]>) => {
    const publicationsList = await Publications.findAll();

    res.json(sendResponse<IPublications[]>(200, 'All good.', publicationsList));
});
```

Εικόνα 4.13: Routers Declaration for Academic Staff Positions

Η sendResponse είναι μία generic μέθοδος που φτιάχνει το response αντικείμενο όπου θα στείλει.

```
export function sendResponse<T>(code: number, description: string, data?: T): ResponseData<T> {
    return {
        code: code,
        data: data,
        description: description,
        success: code === 200,
    }
}
```

Εικόνα 4.14: HellenicCSResearch – Generic Response Object

4.4 Υλοποίηση του frontend

Η διαδικασία υλοποίησης του frontend αποτελεί ένα κρίσιμο στάδιο στην ανάπτυξη ενός προγράμματος ή μιας εφαρμογής, καθώς αποτελεί το παράθυρο μέσω του οποίου οι χρήστες αλληλεπιδρούν με το σύστημα. Στο πλαίσιο αυτού του κεφαλαίου, παρουσιάζουμε τις τεχνολογίες και τα εργαλεία που επιλέχθηκαν για την υλοποίηση του frontend της εφαρμογής μας. Οι βασικές τεχνολογίες είναι η React, η Typescript για την βελτίωση της αξιοπιστίας της εφαρμογής, Redux Toolkit για την διαχείριση του state με έναν απλοποιημένο και αποτελεσματικό τρόπο, Chart.js 2 που έχει react components για την Chart.js και Material-UI (MUI) για την σχεδίαση της διεπαφής.

Η react θα τοποθετήσει το App component στο element με id “root”, που θα είναι το top level component της εφαρμογής.

Αναλυτικά το App component κάνει:

- Provider: είναι μέρος της Redux Toolkit και παρέχει το Redux store σε όλη την εφαρμογή. Το store είναι ο χώρος ο οποίος γίνεται η αποθήκευση του state της εφαρμογής, που περιλαμβάνει τα δεδομένα και τη λογική της εφαρμογής. Θα αναλυθεί περεταίρω πιο παρακάτω.
- ColorModeContenxt: παρέχει ένα περιβάλλον χρωμάτων σε όλη την εφαρμογή. Η μεταβλητή “colorMode” περιέχει το τρέχον χρωματικό σχήμα της εφαρμογής (όπως φωτεινό ή σκοτεινό)
- ThemeProvider: παρέχει ένα θέμα στην εφαρμογή, το οποίο περιλαμβάνει συγκεκριμένες στυλιστικές επιλογές, όπως χρώματα, γραμματοσειρές κ.λπ.
- CssBaseline: το συγκεκριμένο είναι από την βιβλιοθήκη MUI και παρέχει κάποιους κανόνες CSS. Βοηθάει στη διατήρηση της ομοιομορφίας στην εμφάνιση της εφαρμογής.
- Paths: Το βασικό component της εφαρμογής, ορίζει τις διαδρομές (routes) χρησιμοποιώντας την React Router. Προσδιορίζει ποιο component να αποδοθεί για κάθε διαδρομή. Θα αναλυθεί παρακάτω
- TheAlert: Το βασικό alert component της εφαρμογής. Χρησιμοποιείται για να δείχνει ένα μήνυμα στον χρήστη.

```
    return (
        <Provider store={store}>
            <ColorModeContext.Provider value={colorMode}>
                <ThemeProvider theme={theme}>
                    <CssBaseline />
                    <Paths />
                    <TheAlert />
                </ThemeProvider>
            </ColorModeContext.Provider>
        </Provider>
    );
}

2 references
export default App;
```

Εικόνα 4.15: App Component

Το store δημιουργείτε χρησιμοποιώντας την configureStore από την Redux Toolkit. Αυτή η μέθοδος δέχεται ένα αντικείμενο με διάφορες ρυθμίσεις, όπως τον reducer που αναλαμβάνει να ενημερώνει το state της εφαρμογής και περιλαμβάνει τους reducers από το slice. Και τον middleware που δέχεται τους middlewares που έχουν φτιαχτεί με την μέθοδο createApi. Επίσης υπάρχει και ένας custom middleware rtkQueryErrorLogger που διαχειρίζεται τα API που επιστρέφουν το status “FETCH_ERROR”.

```
const store = configureStore({
    reducer: {
        [departmentApi.reducerPath]: departmentApi.reducer,
        [publicationApi.reducerPath]: publicationApi.reducer,
        [academicStaffApi.reducerPath]: academicStaffApi.reducer,
        [yearsRangeApi.reducerPath]: yearsRangeApi.reducer,
        filtersSlice: filtersSliceReducer,
        alertSlice: alertSliceReducer,
    },
    middleware: (getDefaultMiddleware) =>
        getDefaultMiddleware().concat(
            departmentApi.middleware,
            publicationApi.middleware,
            academicStaffApi.middleware,
            yearsRangeApi.middleware,
            rtkQueryErrorLogger
        ),
});
// It will fetch after on internet reconnect or on mount
setupListeners(store.dispatch);

37 references
export type RootState = ReturnType<typeof store.getState>;
2 references
export type AppDispatch = typeof store.dispatch;

2 references
export default store;
```

Εικόνα 4.16: The App Store

Παρακάτω βλέπουμε την δημιουργία ενός slice της Redux Toolkit. Το πρώτο πράγμα που κάνουμε είναι να δώσουμε όνομα στο slice και το αρχικό state του. Μετά ορίζουμε τους reducers που θα έχει και είναι υπεύθυνοι να αλλάζουν το state του slice. Οι παράμετροι που παίρνει ο reducer είναι το τωρινό state του slice και τα καινούργια δεδομένα για να αλλάξει το state.

```
const initialState: IFilterSlice = {
    yearsFilters: { yearsRange: [], unknownYear: false },
    academicPos: [],
    departments: [],
    maxYearsRange: [],
};

const resetState = (yearsRange: number[]): IFilterSlice => {
    const tempInitialState = { ...initialState };
    tempInitialState.yearsFilters.yearsRange = yearsRange;
    tempInitialState.maxYearsRange = yearsRange;
    return tempInitialState;
};
4 references
export const filtersSlice = createSlice({
    name: 'filtersSlice',
    initialState,
    reducers: {
        reset: (state, yearsRange: PayloadAction<number[]>) =>
            resetState(yearsRange.payload),
        setYearsFilters: (
            state,
            payload: PayloadAction<Partial<YearsFilters>>
        ) => {
            state.yearsFilters = {
                ...state.yearsFilters,
                ...payload.payload,
            };
        },
        setMaxYearsRange: (state, maxYearsRange: PayloadAction<number[]>) => {
            state.maxYearsRange = maxYearsRange.payload;
        },
        setAcademicPos: (state, academicPos: PayloadAction<Array<string>>) => {
            state.academicPos = academicPos.payload;
        },
        addDepartment: (state, departments: PayloadAction<Array<string>>) => {
            state.departments = departments.payload;
        },
    },
});
```

Εικόνα 4.17: Filter Slice

Η δημιουργία ενός middleware με το createApi, είναι αρκετά παρόμοιος με του slice. Δίνουμε και εδώ ένα όνομα αλλά η δεύτερη παράμετρος είναι το base url για το endpoint π.χ. localhost. Μετά ορίζουμε τα endpoints που θέλουμε να δημιουργήσουμε. Στο παράδειγμα ορίζεται το query getAcademicPositionTotals, το οποίο δέχεται τον τύπο που είναι GET και το url όπου περνάει παραμέτρους όπως το departments, positions, years και unknown_year.

```
export const departmentApi = createApi({
    reducerPath: 'departmentApi',
    baseQuery: fetchBaseQuery({ baseUrl: import.meta.env.VITE_BASE_URL }),
    endpoints: (builder) => ({
        getAcademicPositionTotals: builder.query<
            ResponseData<IAcademicPositionTotals[]>,
            IFAcademicPositionTotals
        >({
            query: ({ departments, positions, years, unknown_year }) => ({
                url: `${Apis.getAcademicPositionTotals}?${constructQueryString({
                    departments,
                    positions,
                    years: years?.map(String),
                    unknown_year,
                })}`,
                method: 'GET',
            }),
        }),
```

Εικόνα 4.18: Filter Slice

Στο Paths component ορίζονται όλες οι διαδρομές της εφαρμογής. Το TheLayout είναι ένα component το οποίο περιέχει την μπάρα προόδου και την κεφαλίδα (App bar) της εφαρμογής. Όσο αφορά τις διαδρομές:

- "/": εάν ο χρήστης πάει σε αυτήν την διαδρομή θα τον κατευθύνει αυτόματα στην "/citations"
- "/citations" και "/departments": οι κύριες διαδρομές της εφαρμογής. Κάνουν render το "FilterAndDataComponent", το οποίο έχει την μπάρα με τα φίλτρα, τα chips και ανάλογα το path εμφανίζει και το ανάλογο component των citations ή publications.
- "*": εάν καμία από τις παραπάνω αυτές τις διαδρομές δεν ταιριάζει, τότε γίνεται render το component "NotFound" που δείχνει ένα απλό μήνυμα στον χρήστη ότι δεν υπάρχει η διαδρομή που έβαλε.

```
function Paths() {
    return (
        <BrowserRouter>
            <TheLayout>
                <Routes>
                    <Route
                        path="/"
                        element={<Navigate to="/citations" replace />}
                    />
                    <Route
                        path="/citations"
                        element={<FilterAndDataComponent />}
                    />
                    <Route
                        path="/departments"
                        element={<FilterAndDataComponent />}
                    />
                    <Route path="*" element={<NotFound />} />
                </Routes>
            </TheLayout>
        </BrowserRouter>
    );
}

2 references
export default Paths;
```

Εικόνα 4.19: Paths Component

Θα αναλύσουμε το Filter component που υπάρχει μέσα στο FilterAndDataComponent. Επειδή η εφαρμογή ανταποκρίνεται και σε κινητά, έχει γίνει η χρήστη του Drawer component της MUI, το οποίο εάν βρίσκεται σε μεγάλη οθόνη ο χρήστης, θα εμφανίζει την μπάρα με τα filters, ενώ σε κινητά γίνεται μία sidebar η οποία “κρύβεται” γιατί υπάρχει πολύ περιορισμένος χώρος. Αυτό επιτυγχάνεται από την μεταβλητή large που χρησιμοποιεί το custom hook useMediaQuery της MUI και ακούει στην αλλαγή του πλάτους της οθόνης. Μέσα στον Drawer, γίνεται render η μεταβλητή drawer που περιέχει όλα τα components για τα φίλτρα.

Εδώ γίνονται και τα api calls για τα δεδομένα που σχετίζονται με κάθε φίλτρο.

```
return (
    <Drawer
        variant={large ? 'persistent' : 'temporary'}
        open={large ? true : mobileOpen}
        onClose={(event: Event, reason: string) => {
            if (reason === 'backdropClick' && !mobileOpen) return;
            setMobileOpen((currentMobileOpen) => !currentMobileOpen);
        }}
        transitionDuration={large ? 0 : undefined}
        ModalProps={{
            keepMounted: true, // Better open performance on mobile.
        }}
        sx={drawerStyle}
    >
        {drawer}
    </Drawer>
);
```

Εικόνα 4.20: Filter Component

Παρακάτω θα επικεντρωθούμε στο PositionCheckboxes, που είναι ένα από τα τρία Filters που υπάρχουν στην εφαρμογή. Είναι υπεύθυνο για την εμφάνιση ενός συνόλου από checkboxes που αντιστοιχούν σε θέσεις ακαδημαϊκού προσωπικού. Το useUrlParams είναι ένα custom hook το οποίο είναι υπεύθυνο να ελέγχει τις url παραμέτρους αλλά και να ενημερώνει ανάλογα το store με τα καινούργια value. Η λίστα που γυρνάει έχει μία μεταβλητή με τα καινούργια δεδομένα από το url/store και μία μέθοδος η οποία καλείται όταν αλλάζει το state των input. Το useState είναι για να κρατάει τα values από τα checkboxes και να τα ενημερώνει. Το useSearchParams χρησιμοποιείται εδώ για να ενημερώνει άμεσα όταν γίνεται καταργείτε κάποιο checkbox. Η handleCheckboxesChange είναι η μέθοδος που καλείται κάθε φορά που αλλάζουν τα checkboxes και ενημερώνει την checked.

Υλοποιούνται δύο useEffect τα οποία: το πρώτο ακούει στις αλλαγές του checked και καλεί την handleInputChange της useUrlParams που θα ενημερώσει το url και το store, και η δεύτερη που ακούει στις αλλαγές του paramValue για να ενημερώσει το checked.

```tsx
const PositionCheckboxes: React.FC<PositionCheckboxesProp> = ({
    data,
}: PositionCheckboxesProp) => {
    const [paramValue, handleInputChange] = useUrlParams({
        name: ParamNames.AcademicPos,
        data,
    });
    const [checked, setChecked] = useState<Array<string>>([]);
    const [searchParams, setSearchParams] = useSearchParams();

    const handleCheckboxChange = (
        event: React.ChangeEvent<HTMLInputElement>
    ) => {
        const academicPosID = event.target.name;

        if (event.target.checked) {
            setChecked([...checked, academicPosID]);
        } else {
            setChecked(checked.filter((id) => id !== academicPosID));
        }
    };

    useEffect(() => {
        clearTimeout(academicPosTimeout);
        academicPosTimeout = setTimeout(() => {
            handleInputChange({
                checkboxList: checked,
            });
        }, 500);
        // eslint-disable-next-line react-hooks/exhaustive-deps
    }, [checked]);

    useEffect(() => {
        if (paramValue) {
            setChecked(paramValue.split(','));
        } else {
            setChecked([]);
        }
        // eslint-disable-next-line react-hooks/exhaustive-deps
    }, [paramValue]);
```

Εικόνα 4.21: PositionCheckboxes Component

Τα charts είναι αρκετά περίπλοκα στην δομή και λογική τους. Για να κάνουμε render ένα chart, χρειάζεται τις παραμέτρους options και data. Ακολουθεί μια ανάλυση του Academic Staff Positions chart.

Για τα options επειδή πρέπει να ανανεώνονται τα ποσοστά όταν διαγράφεται από το διάγραμμα έπρεπε να ξαναυπολογίζεται. Το component έχει μία useState μεταβλητή που κάθε φορά που διαγράφεται κάποιο label μετράει όλα τα data από τα ενεργά labels. Όσο για το tooltip καλείται μία callback μέθοδος όπου χρησιμοποιεί την μεταβλητή αυτή για να βγάλει το καινούργιο ποσοστό κάθε label.

```
const options = {
    response: true,
    plugins: {
        legend: {
            position: 'top' as const,
            onClick: (e: any, legendItem: any, legend: any) => {
                legend.chart.toggleDataVisibility(legendItem.index);

                setpositionsSum((prevSum) => {
                    return (
                        (prevSum || 0) +
                        (legendItem.hidden ? +1 : -1) *
                            chartsData.datasets[0].data[legendItem.index]
                    );
                });
                legend.chart.update();
            },
        },
        title: {
            display: true,
            text: 'Academic Staff Positions Chart',
        },
        tooltip: {
            callbacks: {
                label: (d: any) => {
                    const percentage = ` ${(
                        +(d.raw / (positionsSum || 0)) * 100
                    ).toFixed(2)}% (${d.raw})`;

                    return percentage;
                },
            },
        },
    },
};
```

Εικόνα 4.22: Chart Options

Για την σωστή λειτουργία του, υπάρχουν τρία useEffects: το πρώτο είναι για να αλλάζει μία useState μεταβλητή που αφορά το color mode της εφαρμογής, το δεύτερο ακούει από το slice store εάν έχει αλλάξει και ανάλογα καλεί το api με τις καινούργιες ακαδημαϊκές θέσεις ενώ παράλληλα κάνει ξανά ορατά όλα τα πεδία του chart. Το τελευταίο ακούει στα δεδομένα που γυρνάει το api και αλλάζει τις labels και positionsSum useState μεταβλητές.

```
useEffect(() => {
    setColorMode(theme.palette.mode);
}, [theme.palette.mode]);

useEffect(() => {
    if (selectedDeps.length) {
        departments({ departments: selectedDeps });
    }

    if (myChartRef.current) {
        const myPie: Chart = myChartRef.current as Chart;

        myPie?.legend?.legendItems?.forEach((legend, index) => {
            if (legend.hidden) {
                myPie.toggleDataVisibility(index);
            }
        });
        myPie.update();
    }
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, [selectedDeps]);

useEffect(() => {
    if (departmentsPositionData?.data) {
        setLabels(departmentsPositionData?.data);
        const newLabels = departmentsPositionData?.data.length
            ? Object.keys(departmentsPositionData?.data[0].positions).map(
                  (position) => position
              )
            : [];

        const positionSums = newLabels
            .map((position) =>
                departmentsPositionData?.data.reduce(
                    (total, department) =>
                        total + (department.positions[position] || 0),
                    0
                )
            )
            .reduce(
                (totalPositions: number, position: number) =>
                    totalPositions + position || 0
            );

        setpositionsSum(positionSums || 0);
    }
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, [departmentsPositionData]);
```

Εικόνα 4.23: PositionsPieChart - UseEffects

Τέλος, για τα δεδομένα του γραφήματος χρησιμοποιείται μια μεταβλητή useMemo. Αυτή η μέθοδος δεν υπολογίζει ξανά τις τιμές σε κάθε ανανέωση (re-render), αλλά μόνο όταν υπάρχουν αλλαγές στις τιμές των εξαρτήσεων που έχουν οριστεί στον πίνακα εξαρτήσεων (dependency array). Αυτές οι εξαρτήσεις περιλαμβάνουν τις ετικέτες (labels), οι οποίες αλλάζουν όταν το API επιστρέφει νέα δεδομένα, καθώς και τον τρόπο χρωματισμού (colorMode), ο οποίος αλλάζει ανάλογα με το θέμα της εφαρμογής.

Το createData χρησιμοποιείται για τη δημιουργία δεδομένων για το γράφημα. Αρχικά, δημιουργεί μια λίστα με τις ετικέτες των θέσεων από τα data. Έπειτα, υπολογίζει τα χρώματα και τα χρώματα περιγράμματος των κελιών του γραφήματος, λαμβάνοντας υπόψη το χρώμα που αντιστοιχεί σε κάθε θέση. Στη συνέχεια, υπολογίζει τον συνολικό αριθμό θέσεων ανά ετικέτα για όλα τα τμήματα. Τέλος, δημιουργεί ένα dataset που περιέχει τα δεδομένα των θέσεων ανά ετικέτα, τα χρώματα φόντου και περιγράμματος και τα επιστρέφει.

```
export const createData = (
    data: PositionsByDepartment[],
    theme: string
): ChartData<'pie'> => {
    const bgColorOpacity = theme === 'dark' ? '0.7' : '0.9';
    const labels = data.length
        ? Object.keys(data[0].positions).map((position) => position)
        : [];

    const colors = labels.map((position: string) => {
        const colorInfo = colorMap.get(position);
        return colorInfo?.backgroundColor.replace('opacity', bgColorOpacity);
    });

    const borderColors = labels.map((position: string) => {
        const colorInfo = colorMap.get(position);
        return colorInfo?.borderColor;
    });

    const positionSums = labels.map((position) =>
        data.reduce(
            (total, department) =>
                total + (department.positions[position] || 0),
            0
        )
    );

    const datasets = [
        {
            label: '# of Academic Staff',
            data: positionSums,
            backgroundColor: colors,
            borderColor: borderColors,
        },
    ];

    return {
        labels: labels || [],
        datasets,
    };
};


const chartsData = useMemo(() => {
    return createData(labels, colorMode);
}, [labels, colorMode]);
```

Εικόνα 4.24: PositionsPieChart – chartsData

Ακολούθως, θα εστιάσουμε στους πίνακες και πιο συγκεκριμένα στον πίνακα των τμημάτων. Στην Εικόνα 2.1 *DeprtmentDataTable Component*, βλέπουμε τον κώδικα που κάνει render το component DeprtmentDataTable. Τυλίγεται από το component ResizableTable το οποίο γυρνάει την παράμετρο height που χρησιμοποιείται στο height του component Paper.

```
<SectionTitle titleText="Departments Statistics" />
<ResizableTable initialHeight={600}>
    {(height) => (
        <Paper
            sx={{
                height: `${height}px`,
                width: '100%',
            }}
        >
            <DataGrid
                className={
                    isDepartmentDataLoading
                        ? 'loading'
                        : 'MuiDataGrid-custom'
                }
                slots={{
                    loadingOverlay: LinearProgress,
                    noRowsOverlay: EmptyData,
                }}
                sx={tableStyle(theme)}
                loading={isDepartmentDataLoading}
                rows={rows}
                columns={columns}
                initialState={{
                    pagination: {
                        paginationModel: { pageSize: 100 },
                    },
                }}
                pageSizeOptions={[10, 25, 50, 100]}
                onRowClick={handleRowClick}
            />
        </Paper>
    )}
</ResizableTable>
```

Εικόνα 4.25: DeprtmentDataTable Component

Τα useEffect που έχει το component, το ένα ακούει σε τρεις διαφορετικές μεταβλητές του store και καλεί το api για τα δεδομένα των τμημάτων. Ενώ το άλλο ενημερώνει τις useState μεταβλητές setTableData και setSelectedDep. Τα useEffect που περιλαμβάνει το συγκεκριμένο component αποτελούν δύο διαφορετικές λειτουργίες. Το πρώτο useEffect ακούει σε τρεις διαφορετικές μεταβλητές του store και εκτελεί το endpoint για να λάβει τα δεδομένα που αφορούν τα τμήματα. Από την άλλη, το δεύτερο useEffect αναλαμβάνει την ενημέρωση των useState μεταβλητών setTableData και setSelectedDep.

Τέλος, υπάρχει και εδώ μία μεταβλητή row useMemo που ακούει στις αλλαγές του tableData, selectedPositions και selectedYears και ενημερώνει τον πίνακα με τα δεδομένα.

```
useEffect(() => {
    if (!isDepartmentDataLoading && departmentsData?.data?.length) {
        setTableData(departmentsData.data);
        if (
            !departmentsData?.data.some((department) => {
                return department.inst === selectedDep;
            })
        ) {
            setSelectedDep(undefined);
        }
    } else if (!isDepartmentDataLoading && !departmentsData?.data?.length) {
        setSelectedDep(undefined);
    }
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, [isDepartmentDataLoading, departmentsData]);

const handleRowClick = (params: GridRowParams) => {
    setSelectedDep(params.id as string);
};

useEffect(() => {
    if (selectedPositions.length && selectedYears.length) {
        departmentsDataReq({
            positions: selectedPositions,
            years: selectedYears,
            unknown_year: selectedUnknownYear,
        });
    }
    // eslint-disable-next-line react-hooks/exhaustive-deps
}, [selectedPositions, selectedYears, selectedUnknownYear]);
```

Εικόνα 4.26: DeprtmentDataTable Component – UseEffects

4.5 Github repository

Το GitHub είναι μια πλατφόρμα φιλοξενίας κώδικα που χρησιμοποιείται κυρίως για τη διαχείριση και τη συνεργασία σε προγραμματιστικά project. Είναι ένας από τους πιο δημοφιλείς και καταξιωμένους χώρους ανάπτυξης λογισμικού στον κόσμο.

Μερικά από τα χαρακτηριστικά του είναι:

- Φιλοξενία Κώδικα: Οι χρήστες μπορούν να ανεβάζουν τον κώδικά τους σε repositories, τα οποία είναι οργανωμένα από projects.
- Έλεγχος Εκδόσεων (Version Control): το σύστημα ελέγχου εκδόσεων Git, το οποίο επιτρέπει στους χρήστες να παρακολουθούν τις αλλαγές στον κώδικα τους, να δημιουργούν διακλαδώσεις (branches) και να συγχωνεύουν τις αλλαγές τους (merge).
- Συνεργασία: Οι χρήστες μπορούν να συνεργαστούν σε project με άλλους χρήστες μέσω pull requests, σχολίων, και αξιολογήσεων κώδικα.
- Ανιχνευτής Αποκλίσεων (Issue Tracker): Οι χρήστες μπορούν να αναφέρουν προβλήματα ή να ζητήσουν νέα χαρακτηριστικά μέσω του συστήματος.
- Διαχείριση Έργων: Οι χρήστες μπορούν να οργανώνουν τα projects τους με οπτικά εργαλεία και πίνακες εργασίας (project boards).

Ο κώδικας της πτυχιακής εργασίας είναι σε ένα monorepo, δηλαδή το frontend και backend βρίσκονται στο ίδιο repository ενώ είναι δύο διαφορετικά project. Το link του repository του project: https://github.com/Petsaki/hellenic-cs-research

Κεφάλαιο 5ο: Παρουσίαση του HellenicCSResearch

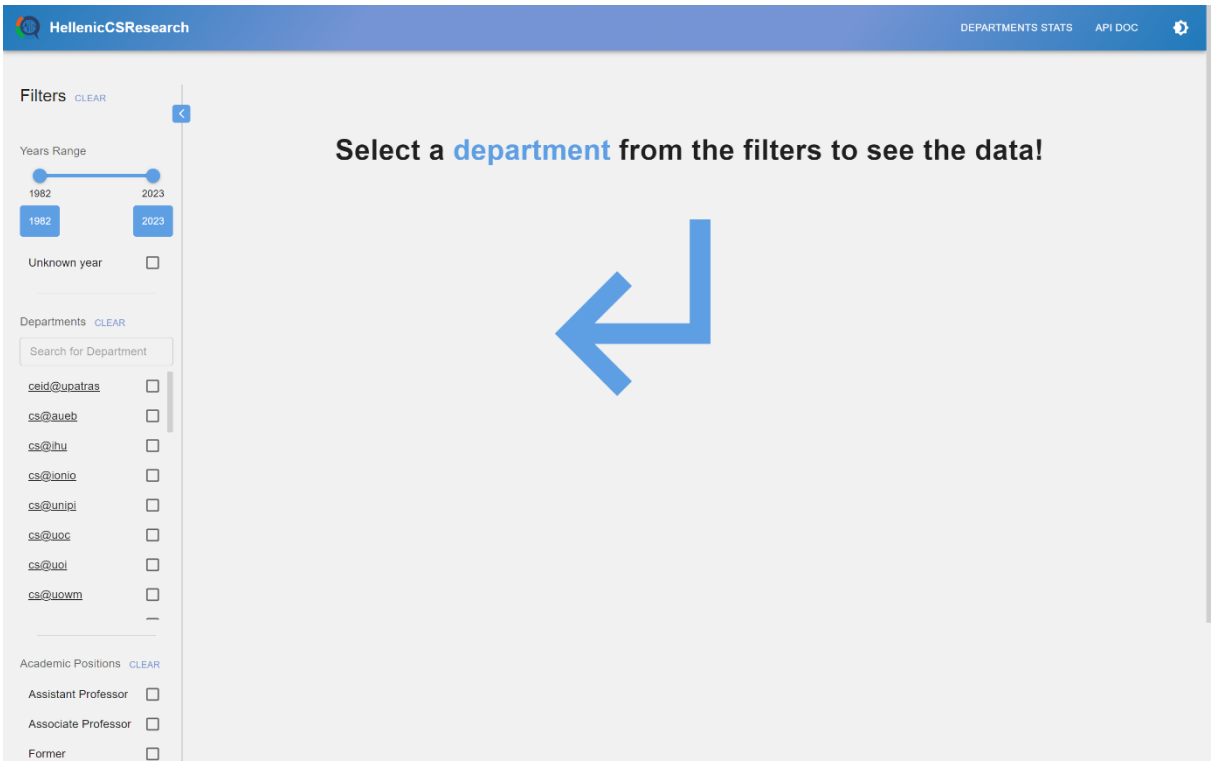
Αυτό το κεφάλαιο αφορά την παρουσίαση και ανάλυση της ιστοσελίδας HellenicCSResearch για την διαχείριση και αξιοποίηση των επιστημονικών δεδομένων, επιτρέποντας στους ερευνητές να παρακολουθούν, να οπτικοποιούν και να συγκρίνουν την επιστημονική δραστηριότητα σε πανεπιστημιακά τμήματα της πληροφορικής της Ελλάδος. Η ιστοσελίδα χωρίζεται σε δύο κύριες σελίδες: ακαδημαϊκό προσωπικό και τμήματα, που θα παρουσιαστούν παρακάτω.

5.1 Αξιολόγηση ακαδημαϊκού προσωπικού

Η σελίδα ενημερώνει τον χρήστη ότι πρέπει να επιλέξει κάποιο τμήμα ώστε να εμφανιστούν τα δεδομένα του εκάστοτε τμήματος ή τμημάτων. Στα Αριστερά υπάρχουν τα φίλτρα τα οποία ο χρήστης μπορεί να επιλέξει και χωρίζονται σε τρεις κατηγορίες:

- Ημερομηνίες για τις δημοσιεύσεις (Years Range): ένας slider με το εύρος χρονολογιών από τις δημοσιεύσεις αλλά και ένα checkbox για την επιλογή εμφάνισης δημοσιεύσεων που δεν έχουν ημερομηνίες δημοσίευσης. Το checkbox είναι ενεργοποιημένο, δηλαδή μπορεί ο χρήστης να το επιλέξει ή όχι, μόνο εάν ο slider έχει ρυθμιστεί στην ελάχιστη και μέγιστη τιμή που μπορεί να λάβει.
- Τμήματα (Departments): Μία λίστα από checkboxes που μπορεί να επιλεχτούν όλα. Λόγω των πολλών επιλογών υπάρχει ένα input για την αναζήτηση τμημάτων και ένα κουμπί clear που θα επαναφέρει όλα τα checkboxes στην αρχική τους θέση. Επίσης υπάρχουν σύνδεσμοι για όλα τα τμήματα εάν γίνει κλικ στο λεκτικό του checkbox.
- Ακαδημαϊκές θέσης (Academic Positions): Μία λίστα από checkboxes που μπορεί να επιλεχτούν όλα. Λόγω των πολλών επιλογών υπάρχει και εδώ ένα κουμπί clear που θα επαναφέρει όλα τα checkboxes στην αρχική τους θέση.

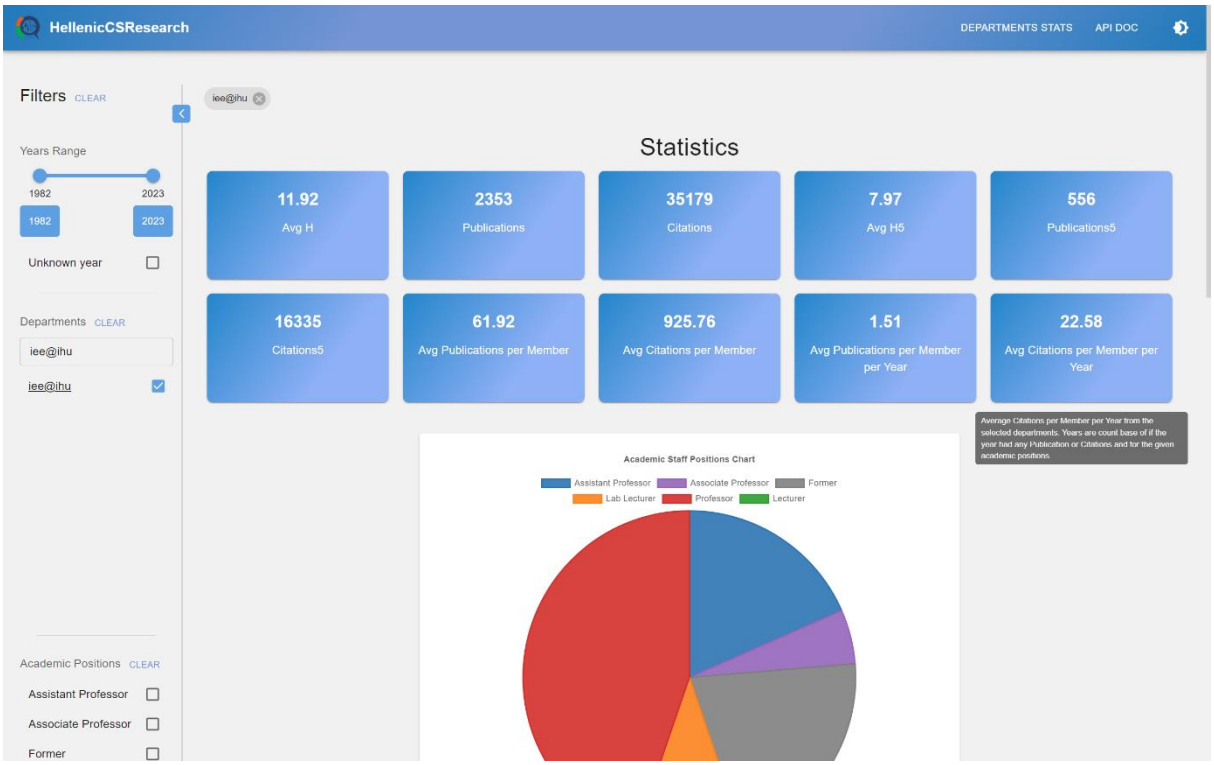
Υπάρχει το κουμπί clear δίπλα από το λεκτικό Filters, το οποίο επαναφέρει όλα τα φίλτρα στην αρχική τους κατάσταση όπως φαίνεται και στην Εικόνα 5.1 *Citation Page without Filters*. Τέλος, δίπλα από αυτό το clear του Filter, είναι ένα ακόμη κουμπί που κρύβει την μπάρα με τα φίλτρα ώστε να γίνεται όσο δυνατόν καλύτερη διαχείριση την οθόνης και η εμπειρία του χρήστη. Δίπλα από την ετικέτα “Filters”, υπάρχει ένα κουμπί clear. Αυτό το κουμπί επαναφέρει όλα τα φίλτρα στις προεπιλεγμένες τους τιμές. Επιπλέον, ακριβώς δίπλα σε αυτό το κουμπί clear υπάρχει ένα ακόμη κουμπί που επιτρέπει την απόκρυψη της μπάρας φίλτρων. Αυτό συμβάλλει στη βελτιστοποίηση της οθόνης και στη βελτίωση της εμπειρίας του χρήστη, καθώς διευκολύνει τη διαχείριση του χώρου εργασίας.



Εικόνα 5.1: Citation Page without Filters

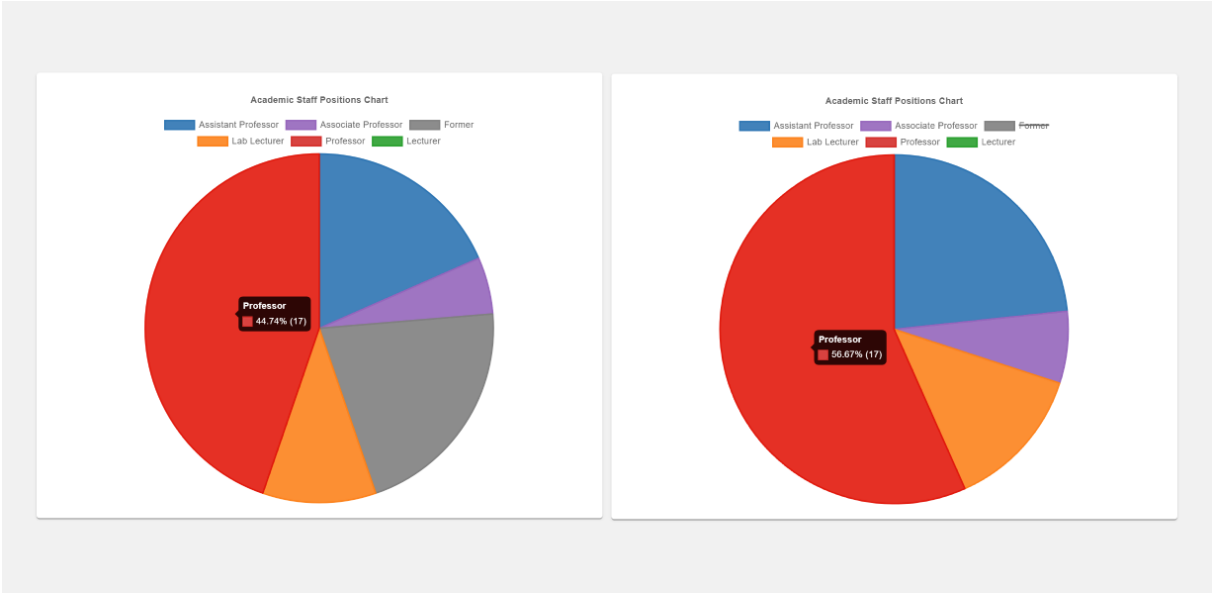
Με την επιλογή κάποιου τμήματος, εμφανίζονται όλα τα στατιστικά δεδομένα και διαγράμματα. Επάνω από τα δεδομένα, υπάρχει το φίλτρο το οποίο επέλεξε ο χρήστης. Μπορεί από εκεί να δει ή και να διαγράψει κάποιο φίλτρο που δεν θέλει.

Υπάρχουν οι κάρτες που δείχνουν τα συνολικά στατιστικά των τμημάτων, στο παράδειγμα της Εικόνα 5.2 *Citation Page with Department filter*, έχει γίνει η επιλογή μόνο ενός τμήματος. Στην περίπτωση που δεν επιλεχτεί κάποια ακαδημαϊκή θέση τότε αυτομάτως η εφαρμογή θα δείξει για όλες τις θέσεις. Μπορεί με την τοποθέτηση του ποντικιού πάνω στις κάρτες, να εμφανιστεί ένα tooltip το οποίο βοηθάει στην κατανόηση των στατιστικών δεδομένων που φαίνονται.



Εικόνα 5.2: Citation Page with Department filter

Στο διάγραμμα Academic Staff Positions Chart, οπτικοποιεί τον συνολικό αριθμό ανά ακαδημαϊκή θέση από όλα τα επιλεγμένα τμήματα. Αυτό το διάγραμμα δεν επηρεάζεται από το φίλτρο Academic Positions. Στην Εικόνα 5.3 *Academic Staff Positions Chart*, δείχνει ότι υπάρχει η επιλογή να διαγράψεις κάποια θέση και να ανανεωθεί το ποσοστό των άλλων θέσεων.

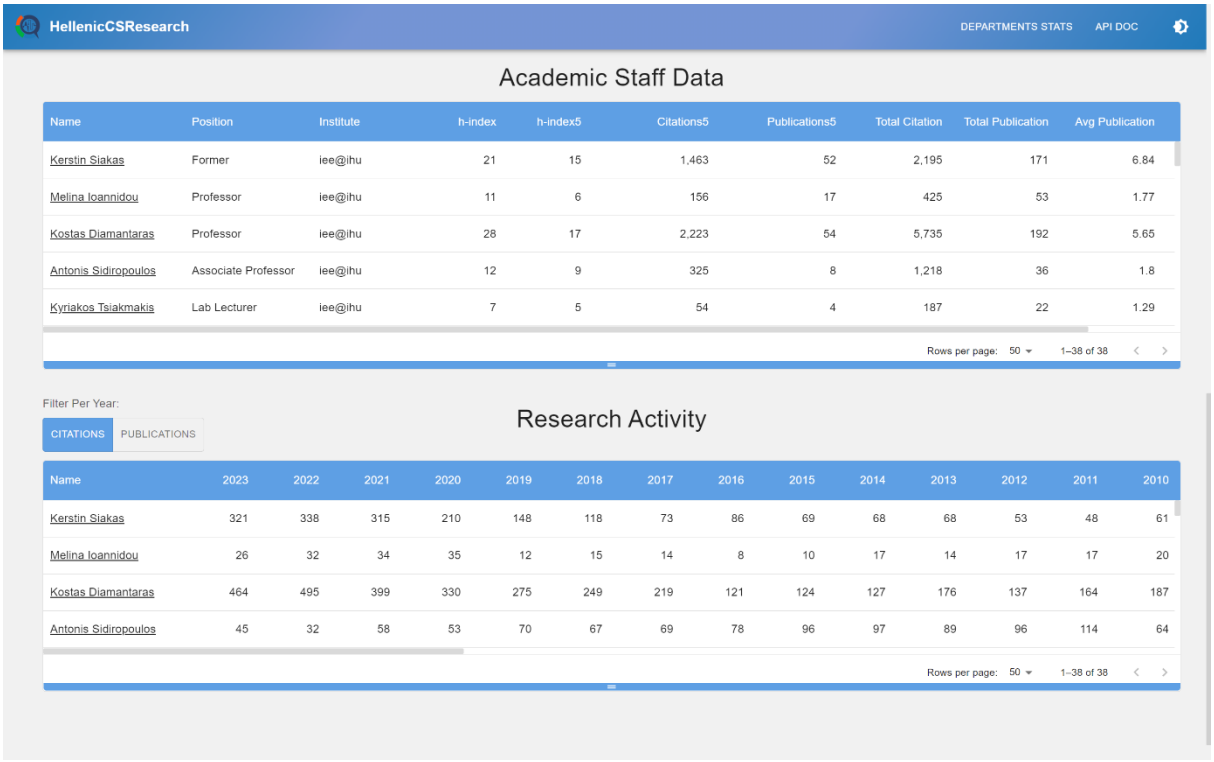


Εικόνα 5.3: Academic Staff Positions Chart

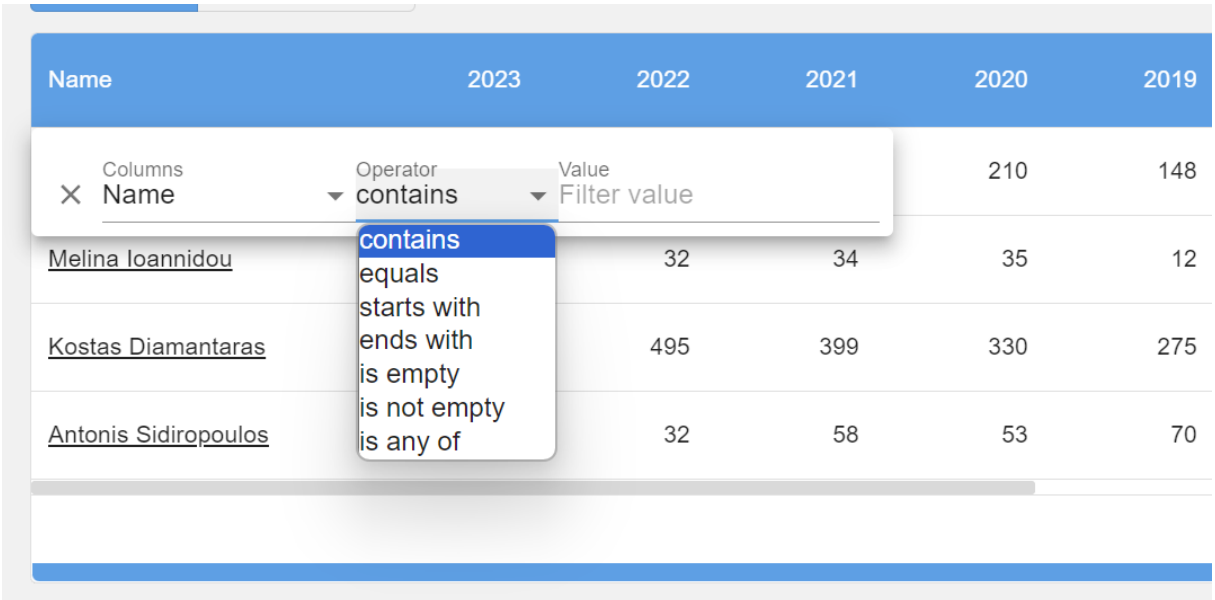
Πηγαίνοντας πιο κάτω στην σελίδα, υπάρχουν δύο πίνακες που δείχνουν δεδομένα με τα σχετικά φίλτρα που έβαλε ο χρήστης. Οι πίνακες είναι resizable στο ύψος, μπορούν να επιλέξουν πόσες γραμμές να εμφανίζει ανά σελίδα και σε ποια σελίδα. Οι πίνακες είναι συνδεδεμένοι μεταξύ τους, δηλαδή εάν αλλάξει σελίδα ή πόσες γραμμές σελίδα ή επιλέξει μία γραμμή, θα γίνει η αντίστοιχη

αλλαγή και στον άλλον πίνακα. Στον “Academic Staff Data” έχει όλα τα στατιστικά στοιχεία του κάθε προσωπικού, ενώ στον “Research Activity” είναι όλες οι επιλεγμένες χρονολογίες και εμφανίζει προ-επιλεγμένα τις παραπομπές ανά χρονιά. Αυτό μπορεί να αλλάξει με τα toggle κουμπιά που βρίσκονται πάνω από τον συγκεκριμένο πίνακα. Υπάρχει η δυνατότητα να εμφανίζει και παραπομπές και δημοσιεύσεις ταυτοχρόνως. Και στους δύο πίνακες υπάρχουν οι επιλογές για αρίθμηση, κρύψιμο μιας στήλης αλλά και φίλτρα τα οποία φαίνονται στην Εικόνα 5.5 *Academic Staff tables - Filter*, για την κάθε στήλη

Επίσης στην Εικόνα 5.4 *Academic Staff tables*, έχει κλείσει η μπάρα με τα φίλτρα.



Εικόνα 5.4: Academic Staff tables



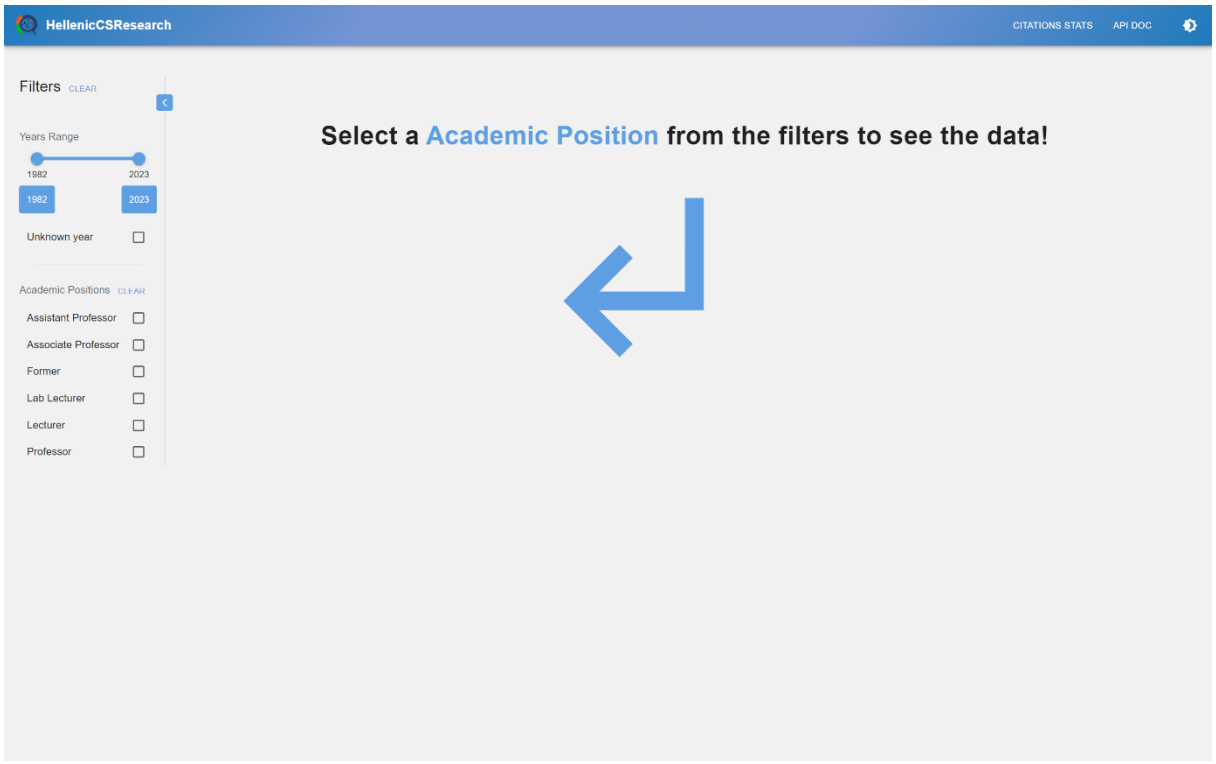
Εικόνα 5.5: Academic Staff tables – Filter

Επιλέγοντας μια γραμμή από τους δύο πίνακες, εμφανίζετε το διάγραμμα με τις παραπομπές και δημοσιεύσεις για τον αντίστοιχο καθηγητή. Στην Εικόνα 5.6 *Research Per Year Chart*, δεν έχει επιλέγει κάποιο από τα δύο toggle κουμπιά με αποτέλεσμα να φαίνονται και τα δύο.

Εικόνα 5.6: Research Per Year Chart

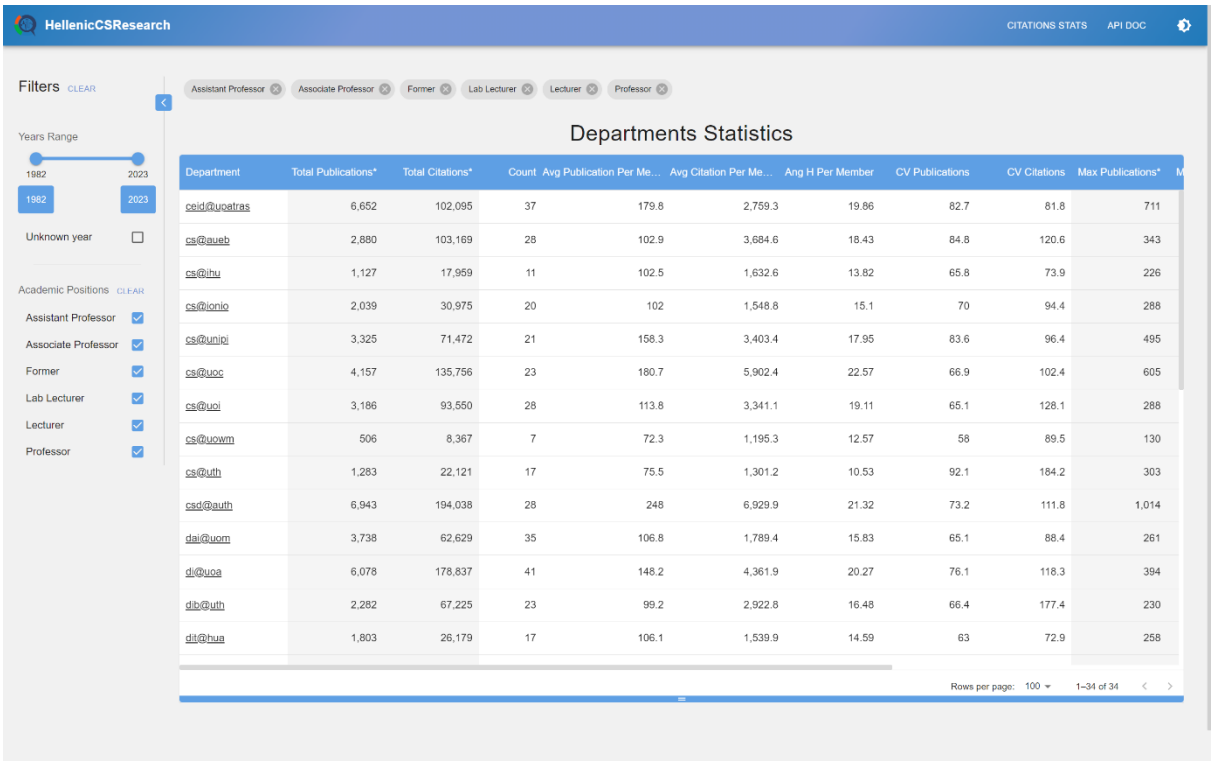
5.2 Αξιολόγηση τμημάτων

Η ιστοσελίδα είναι παρόμοια με την Citation που έχει αναλυθεί στο προηγούμενο κεφάλαιο. Η κύρια και μόνη διαφορά είναι ότι δεν υπάρχει δυνατότητα φιλτραρίσματος ανά τμήματα. Αυτό συμβαίνει διότι η σελίδα αυτή προβάλλει τα δεδομένα για όλα τα τμήματα. Για να εμφανιστούν τα δεδομένα και διαγράμματα θα πρέπει να γίνει η επιλογή κάποιας/κάποιων ακαδημαϊκών θέσεων.



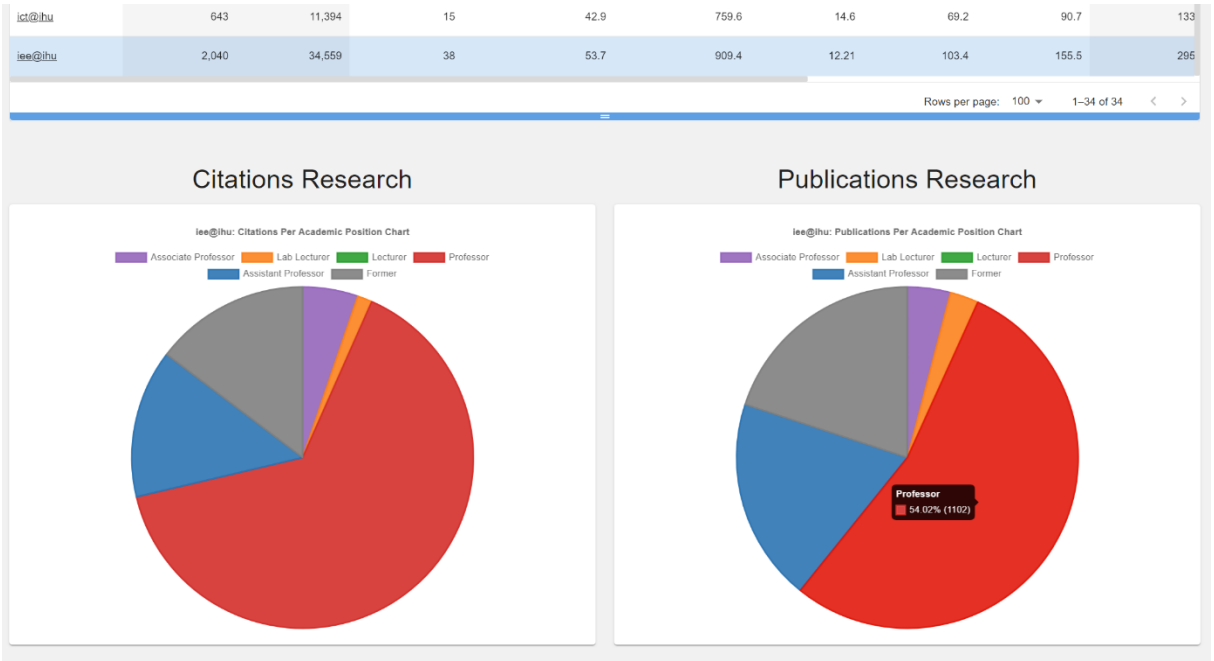
Εικόνα 5.7: Departments Page without Filters

Εφόσον έχουν ενεργοποιηθεί κάποια academic positions φίλτρα τότε εμφανίζεται ο πίνακας με τα στατιστικά στοιχεία όλων των τμημάτων πληροφορικής της Ελλάδος. Οι στήλες οι οποίες έχουν αστερίσκο και γκρίζο χρώμα είναι αυτές που με επηρεάζονται από τα φίλτρα, όλες οι άλλες στήλες τα στατιστικά τους στοιχεία μένουν ίδια. Γίνεται λόγου ότι δεν υπάρχουν αρκετά στοιχεία για να ξέρουμε για παράδειγμα τον συνολικό αριθμό προσωπικού με χρονολογία από το 2000 έως 2010.



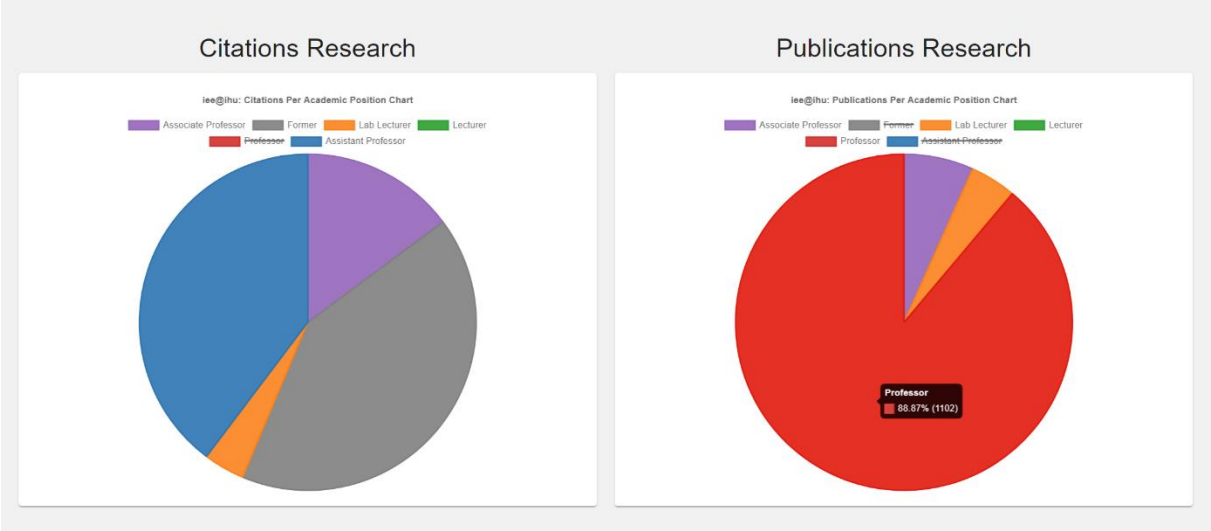
Εικόνα 5.8: Departments Page with Academic Positions filter

Καθώς επιλέχθηκε ένα τμήμα από τον πίνακα, θα προβληθούν δύο pie charts που θα δείχνουν το ένα τις παραπομπές και το άλλο τις δημοσιεύσεις ανά ακαδημαϊκή θέση. Τα charts επηρεάζονται από τα filters. Στην περίπτωση που είχαν διαλεκτοί μόνο τα “Assistant Professor” και “ Associate Professor”, θα εμφανιζόντουσαν μόνο αυτά.



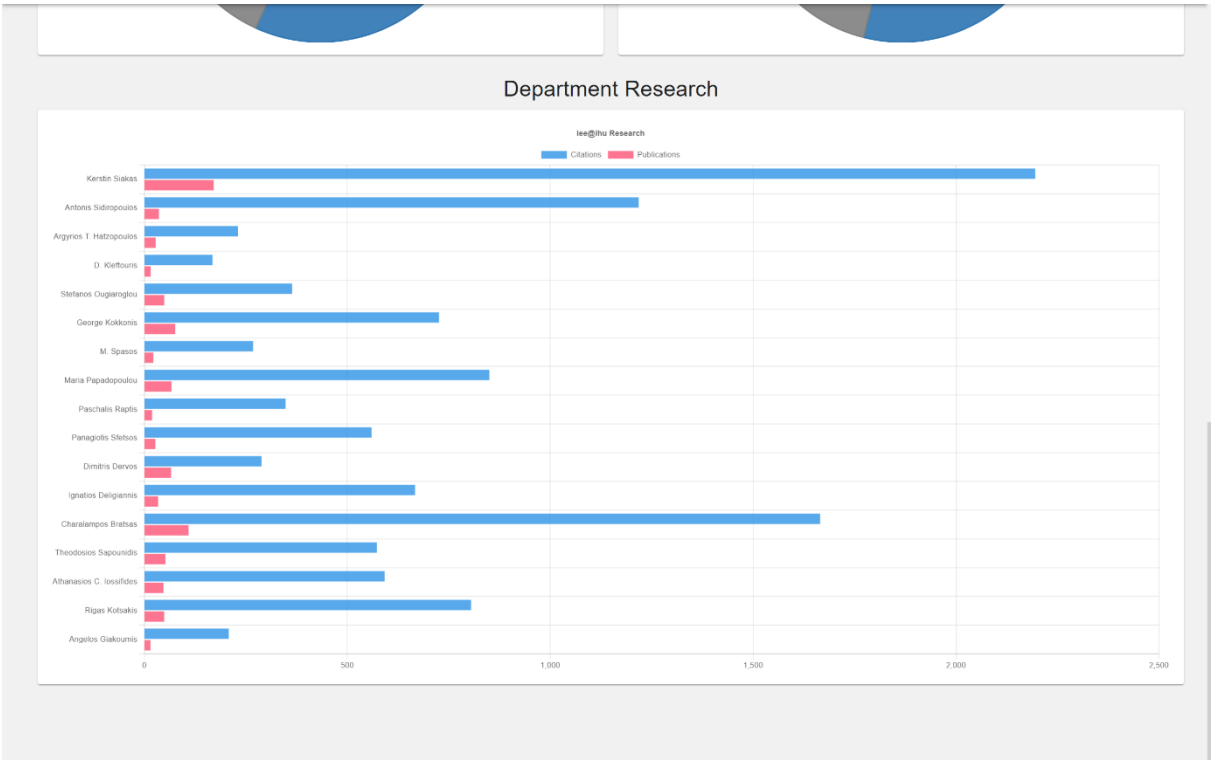
Εικόνα 5.9: Departmental Academic Positions Charts: Citations & Publications

Στην κάτω εικόνα βλέπουμε ότι διαγράφοντας κάποιες θέσης από το διάγραμμα, ανανεώνονται τα ποσοστά.



Εικόνα 5.10: Departmental Academic Positions Charts Filtered Positions - Citations & Publications

Τέλος, υπάρχει το διάγραμμα Department Research, που δείχνει τις παραπομπές και δημοσιεύσεις των καθηγητών με τις επιλεγμένες θέσεις και ανάλογα με το εύρος χρονολογίας.

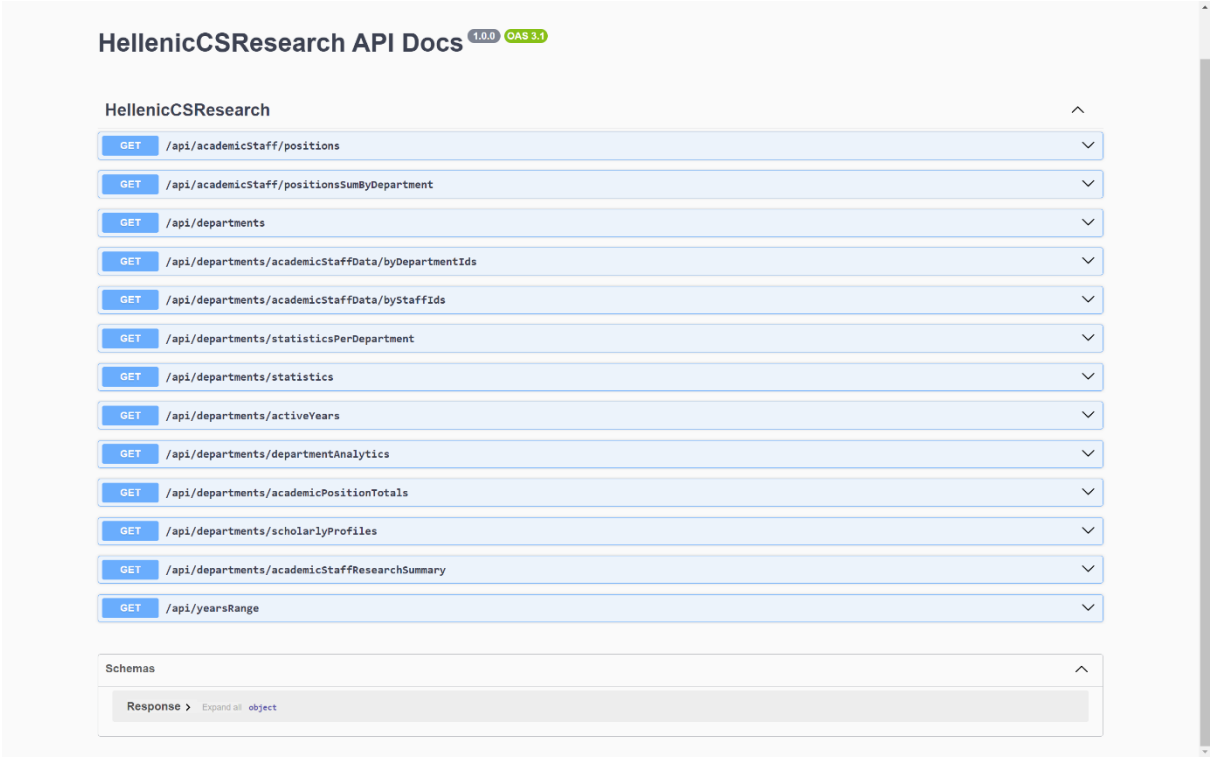


Εικόνα 5.11: Department Research Chart

5.3 Open API

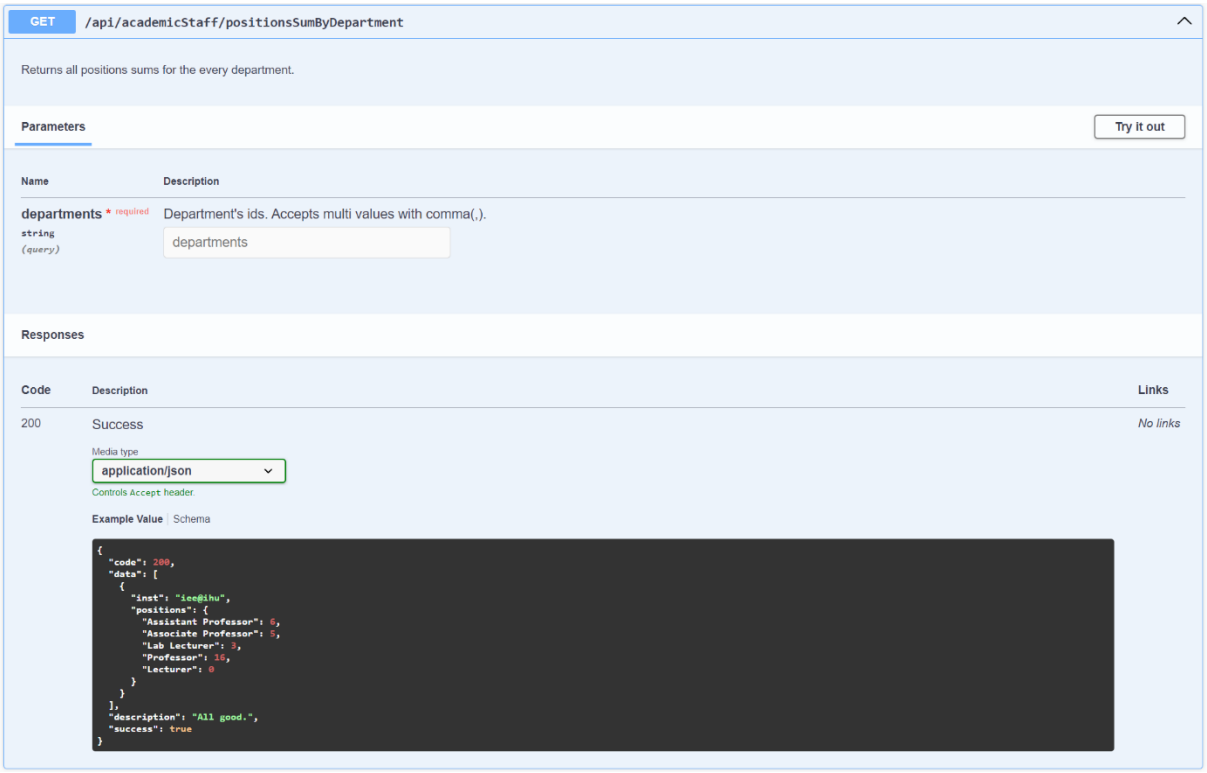
Η εργασία έχει αναπτύξει ένα Open API για την πρόσβαση και διαχείριση δεδομένων. Αποτελεί ένα κρίσιμο εργαλείο που επιτρέπει την επικοινωνία μεταξύ διαφορετικών συστημάτων και εφαρμογών, επιτρέποντας την ανταλλαγή δεδομένων και την παροχή πρόσβασης σε υπηρεσίες μέσω διαδικτύου. Παρέχει μια εύκολη, αποτελεσματική και ευέλικτη λύση για τη διαχείριση δεδομένων, επιτρέποντας

στους προγραμματιστές να αξιοποιούν και να επεξεργάζονται πληροφορίες με άνεση και ακρίβεια. Στην Εικόνα 5.12 *Open API Page*, είναι όλα τα διαθέσιμα apis που προσφέρει η πτυχιακή εργασία.



Εικόνα 5.12: Open API Page

Κάθε endpoint περιγράφεται λεπτομερώς για τι δεδομένα γυρνάει, τι παραμέτρους απαιτεί και ένα παράδειγμα για το τι αποτέλεσμα γυρνάει με κωδικό κατάστασης 200. Επίσης μπορεί να δει όλο το schema το οποίο είναι βασισμένο.



Εικόνα 5.13: Open API – Endpoint

Κεφάλαιο 6ο: Συμπεράσματα και Μελλοντικές επεκτάσεις

6.1 Συμπεράσματα

Στο πλαίσιο αυτής της πτυχιακής εργασίας, εξετάστηκε η ανάγκη για ανάπτυξη ενός API και μιας web εφαρμογής που θα επιτρέπουν την ανάκτηση, οπτικοποίηση και διαχείριση επιστημονικών δεδομένων στον τομέα της Επιστήμης, της Μηχανικής Υπολογιστών, του Ηλεκτρονικού και Ηλεκτρολόγου Μηχανικού. Μέσω της ανάπτυξης ενός μηχανισμού άντλησης και αποθήκευσης δεδομένων από το Google Scholar, επέτρεψε την παρακολούθηση και τη σύγκριση της επιστημονικής δραστηριότητας σε πανεπιστημιακά τμήματα.

Κατά τη διάρκεια της έρευνας, αναγνωρίστηκαν ορισμένες ελλείψεις στην υπάρχουσα διαχείριση και αξιοποίηση επιστημονικών δεδομένων. Παρατηρήσαμε ότι η πρόσβαση σε επιστημονικές πληροφορίες ήταν περιορισμένη και η ανάλυσή τους ήταν συχνά χρονοβόρα και δύσκολη. Επιπλέον, η σύγκριση δεδομένων από διαφορετικά πανεπιστημιακά τμήματα ήταν περίπλοκη λόγω της έλλειψης ενός ενιαίου συστήματος.

Η εφαρμογή κατάφερε να δημιουργηθεί ένα εργαλείο που επιτρέπει στους ερευνητές να αποκτούν πρόσβαση σε σημαντικές πληροφορίες και να διεξάγουν αναλύσεις για την επιστημονική τους δραστηριότητα, παρέχοντας ένα ενοποιημένο περιβάλλον. Μέσω της οπτικοποίησης δεδομένων και των δυνατοτήτων σύγκρισης που παρέχονται, οι ερευνητές μπορούν να αξιολογήσουν την πορεία της έρευνάς τους.

Συνολικά, η πτυχιακή εργασία αποδεικνύει τη σημασία της χρήσης τεχνολογικών εργαλείων στον τομέα της επιστημονικής έρευνας και τη δυνατότητα αξιοποίησής τους για τη βελτίωση της ερευνητικής διαδικασίας και την προώθηση της καινοτομίας.

6.2 Μελλοντικές επεκτάσεις

Η παρούσα έρευνα ανοίγει πολλά πεδία για μελλοντικές επεκτάσεις και βελτιώσεις στον τομέα της διαχείρισης και αξιοποίησης επιστημονικών δεδομένων.

Μια πιθανή επέκταση είναι η ενσωμάτωση νέων πηγών δεδομένων για να επεκταθεί η κάλυψη των επιστημονικών πεδίων. Αυτό μπορεί να περιλαμβάνει την πρόσβαση σε άλλες βάσεις δεδομένων, δημόσια αποθετήρια, ή ακόμα και απευθείας επαφή με ερευνητικά ιδρύματα.

Ένας τομέας που δείχνει σημαντική προοπτική για μελλοντική ανάπτυξη είναι η δημιουργία εργαλείων ανάλυσης δεδομένων. Αυτά τα εργαλεία αποσκοπούν στη δυνατότητα των χρηστών να πραγματοποιούν πιο πολύπλοκες και εξειδικευμένες αναλύσεις από τα συλλεγόμενα δεδομένα, εκμεταλλευόμενα την τεχνολογία της μηχανικής μάθησης και την αναγνώριση προτύπων. Με τη χρήση αλγορίθμων μηχανικής μάθησης, είναι δυνατή η αυτόματη ανίχνευση προτύπων και τάσεων από τα δεδομένα, συμπεριλαμβανομένων της ανίχνευσης ανωμαλιών ή της πρόβλεψης μελλοντικών τάσεων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] L. Bornmann and W. Marx, “The wisdom of studying science from a distance: Scientometrics as a scientific research program,” J. Informetrics, vol. 8, no. 4, pp. 824-837, Oct. 2014.
[2] W. Glänzel and U. Schoepflin, “A bibliometric study of reference literature in the sciences and social sciences,” Inf. Process. Manage., vol. 35, no. 1, pp. 31-44, Jan. 1999.
[3] J. E. Hirsch, “An index to quantify an individual's scientific research output,” in Proceedings of the National Academy of Sciences, vol. 102, no. 46, pp. 16569-16572, Nov. 2005.
[4] L. Egghe, “Theory and practise of the g-index,” in Scientometrics, vol. 69, no. 1, pp. 131-152, Oct. 2006.
[5] B. Jin, et al., “The R-and AR-indices: Complementing the h-index,” Chinese Science Bulletin, vol. 52, no. 6, pp. 855-863, 2007.
[6] J. A. Smith et al., “Citation Patterns in Scientific Software: An Empirical Study,” J. Open Res. Software, vol.4, no. 1, pp. 1-15, 2016.
[7] J. E. Bollen, M. A. Rodriguez, J. Van de Sompel, “Journal status,” Scientometrics, vol. 69, no. 3, pp. 669-687, Dec. 2006.
[8] Moed, H. F. “Measuring contextual citation impact of scientific journals.” Journal of Informetrics, vol. 4, no. 3, pp. 265-277, Jul. 2010.
[9] L. Bornmann and H.-D. Daniel, “What do citation counts measure? A review of studies on citing behavior,” J. Documentation, vol. 64, no. 1, pp. 45–80, 2008.
[10] E. Garfield, “Citation Indexes for Science: A New Dimension in Documentation through Association of Ideas,” Science, vol. 122, no. 3159, pp. 108-111, 1955.
[11] E. Garfield, “The history and meaning of the journal impact factor,” JAMA, vol. 295, no. 1, pp. 90-93, 2006.
[12] Clarivate Analytics. (2021). “Journal Citation Indicators: Percentile in Subject Area.” [Online]. Available: https://clarivate.com/webofsciencegroup/solutions/journal-citation-indicators.
[13] H. Piwowar et al., "The state of OA: a large-scale analysis of the prevalence and impact of Open Access articles," PeerJ, vol. 6, 2018.
[14] L. Bornmann and W. Marx, “The wisdom of studying science from a distance: Scientometrics as a scientific research program,” J. Informetrics, vol. 8, no. 4, pp. 824-837, Oct. 2014.
[15] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab, 1999.
[16] “Scopus Sources”, Scopus 2024. [Online]. Available: https://www.scopus.com/sources
[17] I. Madisch, “A Little Bit of Knowledge Is a Dangerous Thing,” Science and Engineering Ethics, vol. 18, no. 1, pp. 1-3, Mar. 2012.
[18] E. Garfield, “Citation indexes for science: A new dimension in documentation through association of ideas,” Science, vol. 122, no. 3159, pp. 108-111, Jul. 1955.
[19] “Legality and Ethics of Web Scraping”, Researchgate 2024. [Online]. Available: https://www.researchgate.net/publication/352014123_Legality_and_Ethics_of_Web_Scraping
[20] “State of JS Survey, Frontend Frameworks”, Stateofjs 2022. [Online]. Available: https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/
[21] “FaxJs GitHub Repository”, FaxJs GitHub 2024. [Online]. Available: https://github.com/jordwalke/FaxJs
[22] “React Developer Tools”, react.dev 2024. [Online]. Available: https://react.dev/learn/tutorial-tic-tac-toe#react-developer-tools
[23] “React Native GitHub Repository”, React Native GitHub 2024. [Online]. Available: https://github.com/jordwalke/FaxJs
[24] “React Router Documentation”, React Router 2024. [Online]. Available: https://reactrouter.com/en/main
[25] “Chart.js 2 Documentation”, Chart.js 2 2024. [Online]. Available: https://react-chartjs-2.js.org/
[26] “Chart.js Documentation”, Chart.js 2024. [Online]. Available: https://www.chartjs.org/docs/latest

[27] “Redux Toolkit Documentation”, Redux Toolkit 2024. [Online]. Available: https://redux-toolkit.js.org/
[28] “Material-UI is now Material UI”, MUI Blog 2024. [Online]. Available: https://mui.com/blog/material-ui-is-now-mui/
[29] “NPM”, NPM 2024. [Online]. Available: https://www.npmjs.com/
[30] “Node.js Documentation”, Node.js 2024. [Online]. Available: https://nodejs.org/en/docs
[31] “Express documentation”, Express 2024. [Online]. Available: https://expressjs.com/
[32] “Sequelize documentation”, Sequelize 2024. [Online]. Available: https://sequelize.org/
[33] “Zod documentation”, Zod 2024. [Online]. Available: https://zod.dev/
[34] “Swagger documentation”, Swagger 2024. [Online]. Available: https://swagger.io/docs/
[35] “Typescript documentation”, Typescript 2024. [Online]. Available: https://www.typescriptlang.org/docs/